2015-01-01

Neighbor Discovery in Mobile Ad Hoc Networks

Esau Enrique Ruiz-Gaistardo
University of Texas at El Paso, eeruizgaistardo@miners.utep.edu

NEIGHBOR DISCOVERY IN

MOBILE AD HOC NETWORKS

ESAU RUIZ GAISTARDO

Department of Electrical and Computer Engineering

APPROVED:

Michael P. McGarry, Ph.D., Chair

Patricia J. Teller, Ph.D., co-Chair

John A, Moya, Ph.D.

Charles Ambler, Ph.D.
Dean of the Graduate School

NEIGHBOR DISCOVERY IN AD-HOC NETWORKS

By

ESAU RUIZ GAISTARDO, B.S.E.E

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

May 2015

Acknowledgements

I would first like to thank my advisors Dr. Michael McGarry and Dr. Patricia Teller for their invaluable guidance that help me to grow as an individual. Dr. McGarry and Dr. Teller have been more than just academic advisors, they have also served as mentors and guide in personal matters, always willing to provide advice and share their experiences with their students.

I would also like to extend my gratitude to Dr. John Moya for kindly agreeing to serve in my thesis committee.

Table of Contents

Page

Acknowledgements iv

Table of Contents v

List of Tables vii

List of Figures viii

List of Symbols ix

1 Introduction 1

1.1 Problem Statement 1

1.2 Wireless Networks 4

1.2.1 MANETs 7

1.3 Routing Protocols 7

1.3.1 Proactive Protocols 8

1.3.2 Reactive Protocols 9

1.3.3 Hybrid Protocols 10

1.4 Thesis Overview 11

2 Related Work 12

2.1 Improving Performance of a Real Ad-hoc Network by Tuning OLSR Parameters . 12

2.2 MANETS: Performance Analysis and Management 13

2.3 Performance Analysis of OLSR and BATMAN Protocols Considering Link Quality Parameter 14

2.4 Performance modeling of neighbor discovery in proactive routing protocols 15

2.5 Tunning OLSR 15

3 Experimental Plan 18
3.1 Physical Testbed 18
3.1.1 Experimental Description 36
3.2 NS3-based Simulator 37
3.3 Experimental Plan 39

4 Results 41
4.1 Experimental Data 41
4.1.1 Overall Packet Loss Rate 41
4.2 Observations 45
4.3 Analysis 47
4.3.1 Overall Packet Loss Rate 47
4.3.2 Neighbor-flapping 47

5 Conclusions 51
5.1 Summary 51
5.2 Future work 52

Bibliography 53

Curriculum Vitae 57

List of Tables

3.1 Values of ND message interval and ND message hold time with TCI= 120 seconds. 39

4.1 Tuples of (δ, τ) to measure neighbor-flapping. 48

List of Figures

1.1 Neighbor Discovery Process 2

1.2 Trade-off NM interval and NM hold time 4

1.3 Wireless Networks Classification 6

1.4 Routing Protocols Taxonomy 8

3.1 Layout of OLSR hello message format 24

3.2 Layout of TC message 27

3.3 OLSR Multipoint relay 28

3.4 Routing Table 29

3.5 Layout of OLSR general packet format 31

3.6 Node m and n communication schemes 35

4.1 Overall packet loss rate 43

4.2 Overall packet loss rate 2 44

4.3 Node B neighbor-flapping when δ is equal to τ. 46

4.4 Neighbor-flapping topology 48

4.5 Neighbor-flapping comparison 49

4.6 Asymmetric time neighbor-flapping comparison 50

List of Symbols

AODV	Optimal Link State Routing Protocol
CBR	Constant Bit Rate
CTS	Confirm-to-Send
DIFS	Distributed Inter-Frame Space
EIFS	Extended Inter-frame Spacing
FSR	Fisheye State Routing
HSR	Hierarchical State Routing
HDF5	Hierarchical Data Format version 5
ISM	Industrial,Scientific,Medical
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MPR	Multipoint Relays
OLSR	Optimized Link State Routing Protocol
RPGM	Reference Point Group Mobility
RTS	Request-to-Send
SIFS	Short Inter-Frame Space
SNR	Signal to Noise Ratio
STAR	Source-tree Adaptive Routing
TC	Topology Control
TCI	Topology Change Interval
TORA	Temporally Ordered Routing Algorithm
WRP	Wireless Routing Protocol
ZRP	Zone Routing Protocol

Chapter 1: Introduction

1.1 Problem Statement

The focus of this thesis is finding the *neighbor discovery* (ND) parameters that lead to the best performance in mobile ad-hoc networks (MANETs). Specifically, we focus on the performance of different ND message hold time parameters, to be explained shortly. The performance measure we consider is the overall packet loss rate.

MANETs are wireless networks that are formed on the fly and include nodes that are in motion. Since MANETs are formed on the fly and include mobility, they require self-configuration properties. These self-configuration properties include the ability to self-discover the network topology and elect nodes that will server as routers. ND is a process that creates MANETs with the ability to self-discover the network topology by permitting every node to discover other nodes that are within its transmission range (i.e., its neighbors). After nodes discover their local connections to their neighbors, routing protocols can disseminate this local topology information throughout the network. This topology dissemination will allow each node to obtain the full topology information of the MANET. With this information, each node can compute the shortest paths in the network to every other node. This type of routing is referred to as proactive *link state* routing, we will discuss different classes of routing protocols in Section 1.3.

The ND process consists of message exchanges between nodes with the objective of each node discovering all nodes in their transmission range (i.e., their neighbors). As Figure 1.1 shows, the ND process begins when node A broadcasts a *ND message* and node B receives this *ND message* at time *t1*. Node B then updates its neighbor set information to add node A as an *asymmetric* neighbor. At time *t2*, node B broadcasts a *ND message*, in which node A is listed as an *asymmetric* neighbor. When node A receives this *ND message* with itself listed as an *asymmetric* neighbor of node B, node A will update its neighbor set information with node B as a *symmetric* neighbor. This new neighbor set information will be included in the next *ND message* node A broadcasts at time *t3*. When node B receives this *ND message* with itself listed

as *symmetric neighbor*, node B updates its neighbor set information with node A as a *symmetric* neighbor. Finally, the next ND message that node B broadcasts will have node A listed as a *symmetric* neighbor. At this point, the ND process is successfully completed because the two nodes have discovered their symmetric neighbor status. Each node in the MANET *must* follow this ND process to discover all neighbors within its transmission range (one-hop neighbors).

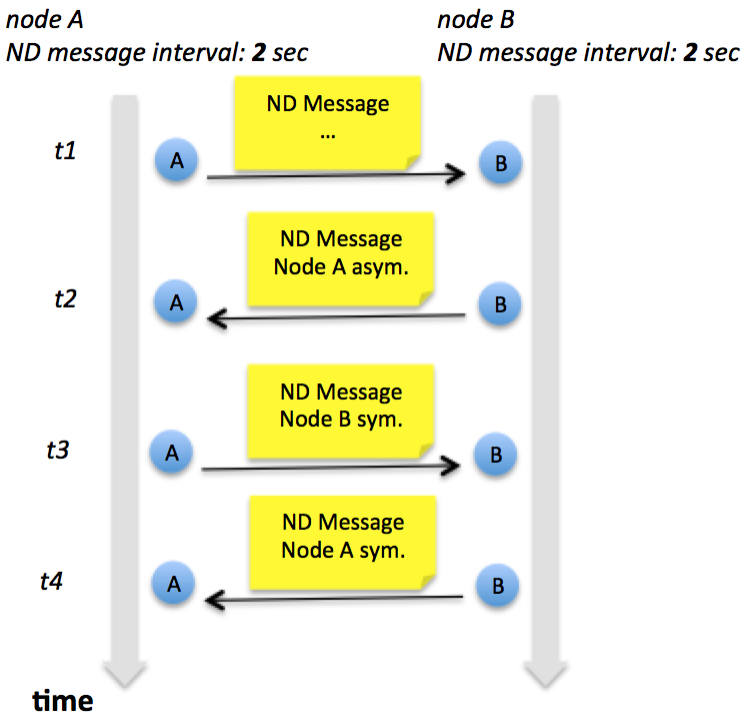


Figure 1.1: Neighbor Discovery Process.

In static networks, the ND process is completed after all *asymmetric* and *symmetric* neighbor relationships are discovered. In MANETs, where the topology of the network changes dynamically, further exchange of *ND messages* is required from each node to maintain up-to-date neighbor status in its neighbor sets. Two parameters govern the ND process: 1) the *ND message interval* (NDM interval), and 2) the *ND message hold time* (NDM hold time). The *NDM interval* parameter determines when ND messages are scheduled to be broadcast [1]. The *NDM hold time* parameter determines the time the neighbor information a *ND message* remains valid. Once the *NDM hold time* expires, the neighbor information provided by the *ND message* becomes invalid

[1].

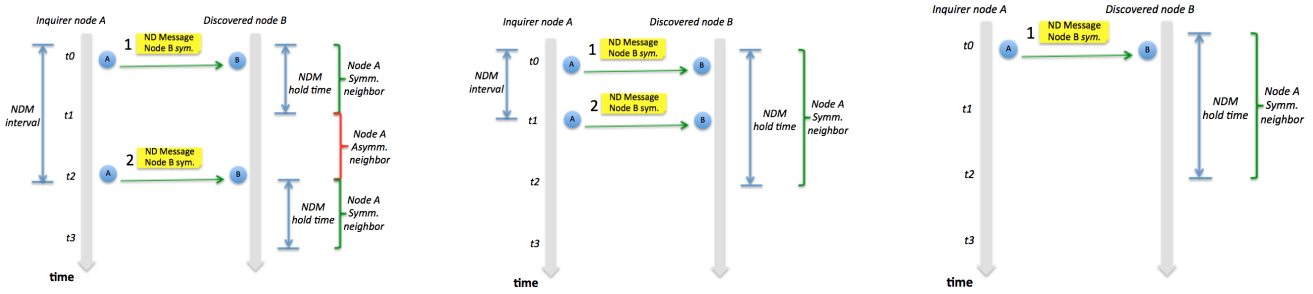
Let's consider two scenarios that occur after node A and $\widehat{B}$ have discovered their symmetric neighbor relationship. The discovery of this relationship occurs after the receipt of a *ND message* at node B from node A at time *t0*.

In the first scenario, nodes A and B maintain their neighbor relationship because neither moves nor changes its transmission range. If the *NDM* hold time <= the NDM interval, as illustrated in Figure 1.2b, then the symmetric neighbor relationship between nodes A and B from the perspective of node B falsely expires, at time *t1*, before the relationship is refreshed by a new *ND message* to be received from node A at time *t2*. Once the *ND message* is received at time *t2*, the symmetric neighbor relationship is restored. These unnecessary oscillations in neighbor relationship are often referred to as neighbor-flapping. If the NDM hold time > the *NDM* interval, as illustrated in Figure 1.2b, then the symmetric neighbor relationship is refreshed at time *t1* by the receipt of the ND message from node A before it was due to expire at time *t2*. Neighbor-flapping, that would cause inaccurate topology information to be disseminated is avoided.

Now consider a second scenario in which nodes A and B move such that they are no longer neighbors at time *t1*. The time it takes from them to discover that their neighbor relationship has been lost is determined by the *NDM* hold time. If the *NDM* hold time expires at time *t2*, then the discovery that nodes A and B are no longer neighbors happens *t2- t1* after the event occurred; see Figure 1.2c. In general, that time is bounded by the *NDM* hold time. During that time period, inaccurate topology information can be disseminated throughout the network leading to misrouting.

To avoid neighbor-flapping, the *NDM* hold time must be greater than the *NDM* interval. However, there is a desire to keep the *NDM* hold time as small as possible to expedite the discovery that two nodes are no longer neighbors. So, the first thought is to set the *NDM* hold time slightly greater than the *NDM* interval, but, How much greater?. In real networks, there are delays transmitting *ND* messages as well as the possibility that *ND* messages are lost. Those

conditions cause neighbor-flapping to occur even if the *NDM* hold time is slightly greater than the *NDM* interval. The objective of this thesis is to uncover some specific guidelines for the setting of the *NDM* hold time using both a physical testbed in ad-hoc mode that we have built with six nodes, and NS3-based simulator that follows the same wireless configuration of the six nodes in ad-hoc mode, as the physical testbed.



(a) NDM hold time $\leq$ NDM interval

(b) NDM interval $<$ NDM hold time

(c) NM hold time with inaccurate information in time t2-t1

Figure 1.2: Trade-off NM interval and NM hold time

1.2 Wireless Networks

Wireless communications use infra-red or radio frequency signals to transmit information between devices [2]. Wireless networks are networks of communication devices that utilize wireless communications with several limitations:

Higher interference results in lower reliability in:

- Infrared signals suffer interference from sun light and heat sources.
- Radio signals are less prone to being blocked, but they are interfered with other electrical devices like microwaves or wireless phones.
- Other devices in the network can interfere with some transmissions.

Low bandwidth available and low transmission rates, typically much slower-speed compared to wired networks, cause degraded quality of service.

Highly variable network conditions produce:

- Frequent disconnections as users move.
- Channel changes as user moves around.
- Poor signal reception with the distance.

Limited transmission resources generates:

- High interference due to medium sharing.
- Limited availability of frequencies with restrictive regulations.
- Spectrum scarce and expensive.

Weak security communications is generated because the radio transmissions are accessible to everyone, therefore network security is more difficult to implement, as attackers can interface more easily [2].

One way to classify wireless networks is based on the number of hops packets cross to reach their final destination (e.g., *one or multiple hops*), and if the network includes infrastructure devices (e.g., MSCs, backbone switches, base stations for 802.11, or cell towers for cellular networks) or not [3]. From these two criteria, we can classify networks as:

- *Single-hop, infrastructure-based* These are the most common wireless networks, all the traffic in the network is administered by a base station. The survivability of the network depends entirely on the base station functionality. See Figure 1.3a.
- *Single-hop, infrastructure-less* In these networks, the base station is removed and replaced by another node to administer the communication in the network in ad-hoc mode, as shown in Figure 1.3b; if the administrator is down, this can be replaced by another node to guarantee the survivability of the network, one example is 802.15.1, a standard based on Bluetooth specifications that works in ad-hoc mode, where nodes organize themselves; these networks require one node to be the master, and the rest of the nodes are slaves; the master coordinates the traffic and the access to the network in TDM (time-division multiplexing) mode.

- *Multi-hop, infrastructure-based* Similar to the Single-hop, Multi-hop infrastructure-based networks use base stations to administer the communication and forwarding data, however some nodes not in transmission range with the base station can forward packets to other nodes to communicate with the base station, a typical application of these networks are sensor wireless networks. See Figure 1.3c.
- *Multi-hop, infrastructure-less* These are self-organizing and self-configuring ad-hoc networks, where nodes act as end-points of a data interchange or as routers. See Figure 1.3d.

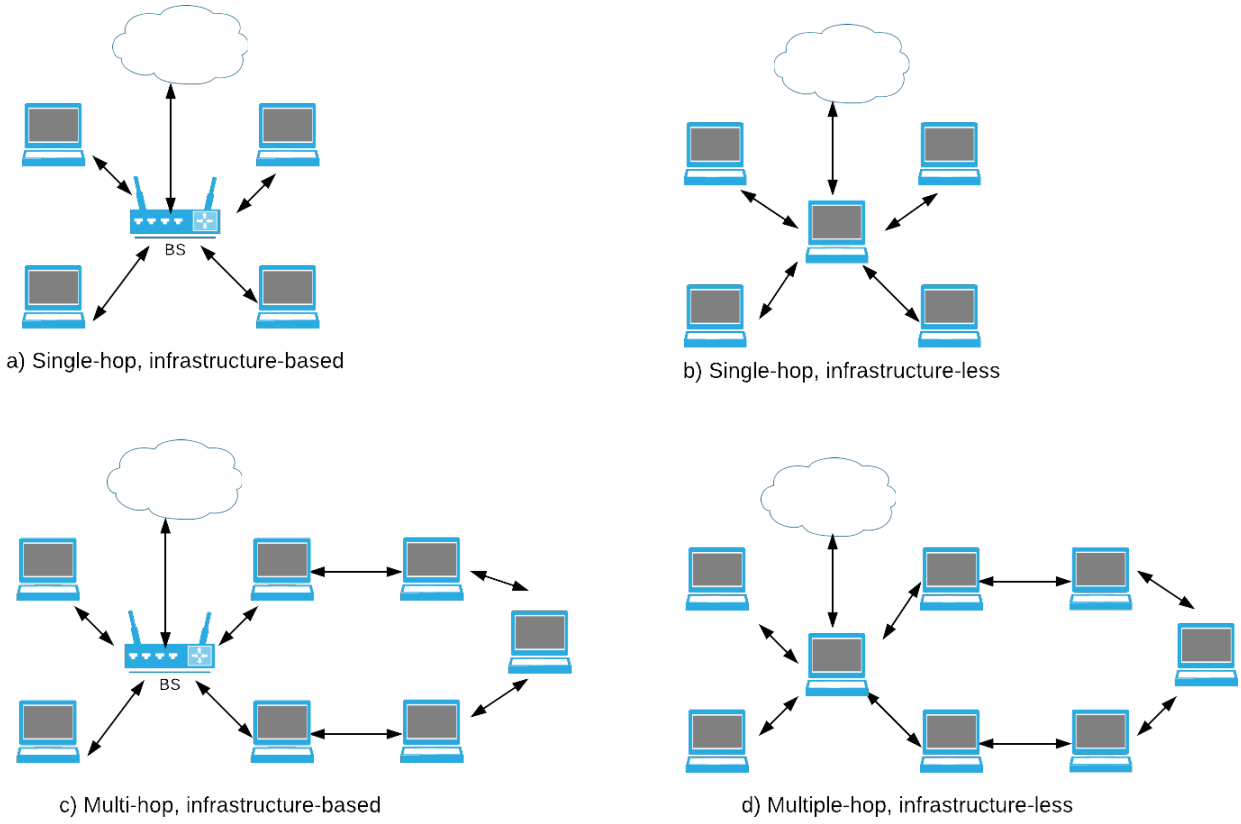


Figure 1.3: Wireless Networks Classification.

1.2.1 MANETs

MANETs are self-organizing networks suitable for different applications (e.g., collaborative computing, disaster recovery, and battle field communications); MANETs bring several benefits over structured wireless network protocols. They are rapid and easy to deploy, improve *flexibility* and *mobility*, and reduce cost. Another distinction for MANETs is they can easily adapt to hostile environments [4]. In MANETs, the process to construct the end-to-end communication between a sender that needs to locate the receiver or receivers begins with the ND process that dynamically advertise of one-hop physical location in the network, this information is used to create forwarding and routing algorithms used to route information through the MANET [5]. Therefore, reliable data transmission in the MANET depends on the accuracy of the ND processes.

1.3 Routing Protocols

Routing protocols are responsible for disseminating topology information and computing shortest paths. Several *routing algorithms* have been developed to compute shortest paths, but only two are used in practice: *link state* and *distance vector* [3]. Routing protocols based on *link state* algorithms, disseminate topology information, so each node has a picture of the network to facilitate the computation of the shortest path, then the forwarding tables are setup to determine the next hop for each packet. For routing protocols base on *distance vector* algorithms, nodes exchange their neighbor set information to facilitate the computation of shortest paths.

Requirements on adaptability and network conditions such as size, traffic density, topology changes, and network partitioning, are additional constrains to consider for routing protocols design [4] [6]. Different routing protocols have emerged to solve aforementioned requirements; these protocols can be classified based on the role nodes play in the network in *uniform or flat* and *hierarchical or hybrid* protocols. Uniform routing protocols use routing tables that have entries on all nodes. Each node in the MANET keeps information of the rest of the nodes in the network. This scheme is convenient for MANETs with low number of nodes but increases

overhead of the control traffic as the network size increase. Hierarchical networks are organized in *clusters* or *zones*; each of these clusters and zones need a cluster or zone leader to maintain nodes' membership information. Hierarchical protocols provide better performance than uniform protocols for MANETs with higher scalability and mobility requirements [4] [5] [6].

Another way to classify routing protocols is based on the frequency of routing information exchange. Based on this criteria, routing protocols can be classified in *Proactive*, *Reactive* and *Hybrid or Hierarchical* Protocols; as Figure 1.4 shows, uniform routing protocols are classified in *Proactive* protocols that periodically disseminate the routing information and *Reactive* protocols, where the status of all nodes is disseminated on demand or when the routes are needed through the network. Hierarchical routing protocols are proactive and reactive in nature, and can be classified in *Zone Hierarchical Routing* and *Cluster-based [7].*

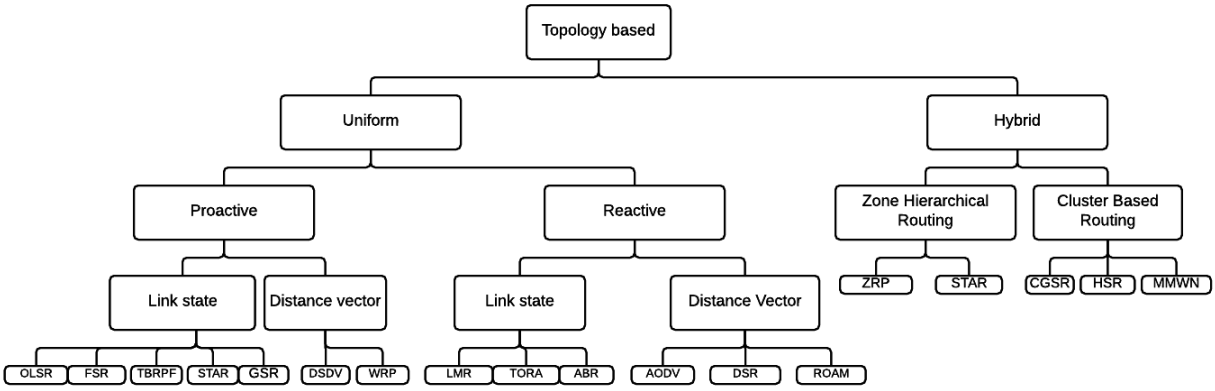


Figure 1.4: Routing Protocols Taxonomy.

1.3.1 Proactive Protocols

Proactive protocols are referred to as global protocols [7] or table driven protocols [6]. In MANETs using a proactive protocol, each node disseminates routing information to every node regardless if the node needs the routing information. This information is maintained in forwarding tables and is periodically updated as routes change. If any topology change happens, the information rapidly propagates to all nodes. This methodology followed by proactive protocols

produces higher routing efficiency in terms of topology update, but this up-to-date routing information scheme creates scalability limitations because the control traffics' overhead grows in the order of $O(N^2)$, creating excessive traffic in the transmission channel [7]. Examples of proactive protocols are: *Destination-Sequenced Distance Vector* (DSDV) [8], *Optimised Link State Routing* (OLSR) [9], *Fisheye State Routing* (FSR) [10], and *Global State Routing* (GSR) [11].

1.3.2 Reactive Protocols

Reactive Protocols, also known as On-Demand Protocols were designed to reduce the overheads in proactive protocols by maintaining information of active routes only, each mobile node operates as specialized router. Routes are obtained as needed and nodes do not require to disseminate periodic routing advertisements, thereby diminishing congestion problems in the network and at the same time reducing overhead control information in the networks. On-Demand Protocols typically initialize with a route discovery phase algorithm. As a consequence, query packets are flooded in the network in search of the path and finish when the route is found or when all possible paths from source have been visited. As mobility increases, longer delays are expected in the routing discovery process, affecting the scalability of the network. Examples of reactive protocols are: *Ad-hoc On Demand Distance Vector* (AODV) [12], *Dynamic Source Routing* (DSR) [13], and *Signal Stability Adaptive* (SSA) [14].

Reactive Protocols have two routing schemes: *source routing* and *hop-to-hop routing* [7]. In *source routing*, each data packet carries the complete source to destination address overhead. Therefore, each intermediate route node reads from packet where to forward it. This scheme brings a drawback in large networks because they do not work well for two main reasons. Firstly, as the number of intermediate route nodes increases, the probability of failure proportional increases. To show this, let $P(fail) = a \times n$ where P(fail) is the probability of route failure, a is the probability of a route node fails, and n the number of route nodes, and $n \to \infty$, $P(fail) \to \infty$. Secondly, as the number of nodes in the route increases the packet overhead, and the probability of delivery failure increases as well [6].

In *hop-to-hop* or *point-to-point* routing, each data packet only carries the destination and next hop address saving packets overhead. In this scheme, each intermediate node, maintains a routing table that needs to maintain an up-to-date routing information for each active routing. Each node may require to be aware of their surrounding neighbors through the use of beacon messages for Neighbor Discovery (ND).

1.3.3 Hybrid Protocols

Proactive Protocols and On Demand Protocols have certain merits and drawbacks. Hierarchical routing protocols combine the benefits of these two type of protocols. They are organized in clusters or zones, whereas nodes use proactive routing protocols to communicate with nodes within their zone or cluster and use on demand protocols to maintain communication with nodes outside their cluster or zone. Hierarchical protocols are a new generation of routing protocols that reduce the broadcasting or control traffic and increase scalability [6]. In hierarchical routing protocols, some nodes are assigned managerial and routing functions. The logical arrangement is to group geographical close mobile nodes in a logical cluster, where one node is assigned to be the cluster-head to communicate other heads with the cluster [7].

Another scheme of hierarchical protocols is where nodes are divided in geographical zones; those inside a zone only carry small information of the nodes in the same zone, to communicate with the nodes outside the zone. Some nodes act as gateways for inter zone communication.

Zone hierarchical routing protocols follow a hybrid scheme where nodes inside the zone communicate using proactive protocols to maintain the status of the nodes inside the zone, and some nodes that act as gateways for inter zone communication use on-demand protocols to communicate with nodes in other zones. Examples of hierarchical protocols are: Hierarchical State Routing HSR [15], Cluster-head-Gateway Switch Routing CGSR [16], Zone Routing Protocol ZRP [17] and Landmark Ad Hoc Routing Protocol LANMAR [18],

1.4 Thesis Overview

This thesis makes the following contribution to the research literature:

1. provides a comprehensive understanding of the ND message hold time parameter values that provide the lowest overall packet loss rate that emerges from mis-routing due to incorrect topology information.

2. uncovers the ND message hold time that provides the lowest overall packet loss rate and the reason why.

These contributions are related to the existing literature on ND parameter values in Chapter 2 to clearly delineate the contribution of this thesis. This thesis is a continue work to measure the ND parameters to find the optimal values in a MANET; to accomplish this goal, we evaluated different ND message interval and ND message hold times values to minimize the overall packet loss rate. We extended our experiments to measure the neighbor-flapping phenomenon. Following the same procedure based on the physical testbed and the NS3-based simulator; we run a set of experiments with different ND parameters values to quantify the neighbor-flapping.

The rest of this thesis is organized as follows: In Chapter 2 a discussion is presented of the existing literature on both ND in general and specifically on ND parameter values. In Chapter 3 details are presented on the experimental platforms utilized and the experiments that were conducted. In Chapter 4 we present the experimental results and their analysis is presented and discussed. Finally, Chapter 5 the thesis is concluded with a summary of the findings and outline of avenues for future work.

Chapter 2: Related Work

In this chapter, we review related studies that have been published about neighbor discovery based on OLSR routing protocol and describe the procedure followed in their analysis, the metrics of performance these papers measure, and review the difference with our study.

2.1 Improving Performance of a Real Ad-hoc Network by Tuning OLSR Parameters

In their paper, Gomez and Garcia, et al., [19] measure the performance of the MANET in two avenues. In the first, the authors calculate and measure the time a node will last to discover or Route Change Latency (RCL) a new route when an intermediate node or MultiPoint Relay (MPR) is no longer in transmission range and the sender needs to find an alternative path by choosing a different MPR to maintain communication with the receiver node. The study consist of calculating the minimal RCL based on the time a node detects that the intermediate node or MPR is not a neighbor and the time the node finds another MPR to re-establish communication with the receiver. The experiments are designed with 4 different sets of ND and TC (topology control) parameters in the OLSR protocol, and using laptops as end-points and PDA as MPRs, all nodes used devices with WiFi technology for communication; and use UDP flow of data packets in a CBR with intervals of 2 msec. The RCL is measured in the receiving.

The 4 sets of experiments use ND message intervals of 0.5 sec, 1 sec, 2 sec and 4 sec; in all cases the ND message hold time is 3 times the value of the ND message interval. The TC interval is 2.5 times the Hello interval; and, TC hold time is 3 times the TC interval. The authors report the minimal RCL delay when the ND message interval is 0.5 sec, and increases as the ND message interval is 4 sec. The physical experiment results are comparatively higher than the minimum expected calculated. This study reports a higher delay of RCL on average 7 min the physical experiments compared with the values calculated. The authors do not report any consideration of noise measurement when run the physical experiments that may provoke the

difference in the results calculated and the results obtained in the physical experiments.

2.2 MANETS: Performance Analysis and Management

In [20], Demers and Kant, et al., use the discrete time simulator OPNET to simulate 4 different network configurations. In the first configuration, the authors installed two wire-connected routers using Route Information Protocol (RIP) for communications and communicating with the first router; a sender node communicates with the first router using OLSR as routing protocol; and a receiver server communicates with the second router using OLSR as routing protocol. In the second configuration, the authors added two MPR nodes, one MPR between the sender node and the first router, and the second MPR communicating the second router and the receiver server. In the third configuration, two more MPRs were added, one communicating to the first router and the first MPRs and the other MPR between the second router and the second MPR communicating the receiver. In the last configuration, they removed the last 2 MPRs and added more nodes in each subnets (in the sender side and the first router and in the receiver and the second router). The data traffic was simulated broadcasting voice streaming in a CBR, each experiment lasts 60 minutes with around 150 to 200 seconds of data setup.

In their work, the authors measure the impact in overhead vs. routing convergence time using *ND message intervals* of 0.5, 1.0, 2.0, 4.0, 6.0, 8.0 and 10.0 seconds (No *ND message hold time* value configuration change is confirmed), and with a *TC* messages interval fixed in 5 sec. In a second set of experiments, the authors configured *TC* messages intervals to 0.5, 1.0, 2.0, 4.0, 6.0 and 8.0 seconds (no *TC* messages validity configuration change is confirmed), and fixed *ND Interval* to 2 seconds.

In [20], the authors report that minimal change overhead and route convergence time when *NDInterval* change compared with the experiments in which topology control parameter changes. The authors conclude that *TC* messages provide higher overhead than *ND messages*, and the route convergence time when a MPR is removed is comparative higher than the time a non-MPR node is removed from the network. In their work, the authors do not report statistics of

the network performance.

2.3 Performance Analysis of OLSR and BATMAN Protocols Considering Link Quality Parameter

In [21], Barolli and Ikeda, et al., run experiments in a physical test-bed with 5 nodes and a gateway. All 5 nodes are maintained within transmission range during all the time experiments last. In this study, the authors work with 3 different levels of mobility in a MANET constructed based on OLSR routing protocol. The nodes are physically distributed in a building following a bus topology or linear distribution.

In the first set of experiments, all nodes remain stationary; in the second set of experiments, the first node in the MANET moves from its original position away from its closest neighbor maintaining the same bus topology and transmission range with its closest neighbor; in the third scenario, the same node continue moving in following the same path along with an intermediate node that follows the same linear movement with the first moving node, and return to their own original position. The movement lasts for 10 sec and is repeated until 50 cycles are completed. The data traffic is generated using a Distributed Internet Traffic (D-ITG) generator with a transmission rate of 122 pkt/s and packet size of 512Kb. The experiment was designed to send the packets with UDP protocol; a second experiment run using TCP protocol. The authorâĂŹs interest is to measure throughput, end-to-end delay, jitter, and packet loss. Neighbor Discovery and Topology Control parameters are not modified during the experiments. The authors designed their experiments based on changing LQWS10 (*link quality window size*) of 10, LQWS100, where the tolerance of the number of lost control packets (ND messages or TC messages) increases with the link quality size without losing the symmetric status. All experiments implemented B.A.T.M.A.N, which solves the problem in OLSR protocol of routing loops created in the routing tables due to *TC* messages and routing table mis-synchronization.

The results provided are measured from one of the fixed intermediate nodes, letting the rest of the four nodes results uncovered. For packet loss do not reflect significant increase for the

experiments for one and two mobile nodes packet loss in the short period of time that experiments run, in order to obtain more reliable results, our experiments run for a long period of time to reflect more reliable performance measurements. Finally, the authors conclude that using OLSR as routing protocol with TCP dataflow, improves the throughput, RTT, jitter, and packet loss when the LQWS is reduced. In this study, the authors use a physical testbed to measure different user-performance metrics such as throughput, end-to-end delays, jitter and packet loss with limited mobility. In our set of experiments, we can simulate a higher level of mobility, however the novelty of this work is that they evaluate the performance using the B.A.T.M.A.N protocol that we can extend for performance evaluation in future works.

2.4 Performance modeling of neighbor discovery in proactive routing protocols

Dynamic ND process addresses different errors, Medina and Bohacek et al. in [22], classify two types: Type I and Type II. Type I error occurs when *NM hello hold* is too long to notice when a neighbor has moved away, but the node still considers the neighbor as *symmetric neighbor*; error Type II happens when the setting of *NM interval* is too long, that loses accuracy when a node is not capable to detect a neighbor within transmission range. Medina and Bohacek highlight Type II as a critical problem, because it reduces connectivity in the network with unproductive links, and diminishes its performance. Aforementioned errors are eliminated with optimal configuration of the ND parameters. Medina and Bohacek et al. also evaluate a link flap effect, and complete an analysis based on mobility rather thank the ND message hold time, as we can see in our experiments results, the link and neighbor flapping problem can be eliminated varying the ND message hold time.

2.5 Tunning OLSR

In [23], Huang and Bhatti, et al., evaluate the performance of OLSR protocol in terms of the throughput and overhead with different ND Intervals, understanding by throughput as the data traffic transferred in a period of time, for data traffic, the simulator generates a constant bit

rate CBR with packet size fixed of 512 bytes, and each node is sender and receiver of all other neighbors in the network; for overhead, the authors measured the number of ND messages and TC messages transmitted during the experiments, and the number of bytes in the routing packets transmitted, finally. The authors calculated the normalized routing overhead, NRO or the ratio of control traffic vs. data traffic.

The experiments were conducted in the discrete time simulator NS2, where the control variables used was mobility from 0 to 30 m/s, with a Δ =5 m/s; (the length of the experiments are not specified). The ND message interval values used are: 1 sec, 2 sec, and 3 sec, and in all experiments, the ND message hold time was 3 times the ND message interval, and the TC interval was settled to 5 sec.

In the experiments, the mobility of the nodes was smoothly increased in an area of 1000 squared meters, the nodes density was settle to 20 for experiments with low traffic, and 50 nodes for experiments with high traffic. The authors report high average throughput in the 3 different ND values measured with low mobility and the throughput reduces as the mobility increases, throughput experience maximum reduction when the ND message interval is set to 3 sec; this behavior is observed for low and high nodes density.

There was a higher impact of NRO when the ND message interval is set to 1 sec, and this value reduces as the ND message interval increases; in the case of throughput, the behavior is the same for high neighbors density and low neighbors density.

To continue with this study, the ND message interval is used as the control parameter varying from 1 sec to 10 sec with a Δ = 0.2s. with a ND message hold time ratio of 3 times the ND message interval, with two different conditions of mobility: with only two levels of speed: low speed of 5 m/s and high speed of 20 m/s, and the two conditions of neighbor scale of small density of neighbors of 20 nodes, and high density of neighbors with 50. In the case of throughput, it decreases almost linearly as the ND message interval increases, the major impact of decay is with higher mobility and higher node density.

NRO decreases as the ND message interval increases; but the decay rate is almost the same

for high and small nodes density. The experiments were repeated 10 times, and the average is presented as the experiments results. The experiments run with fixed values of ND message intervals of 1 sec, 2 sec and 3 sec, and varying the TC interval from 0 sec to 10 sec in intervals of 0.2 sec.

Finally, the authors conclude that the throughput performance reduces when the ND message interval increases, and not impact is reported in the experiments when TC interval varies. On the other hand, NRO has more impact when TC interval varies rather than when ND message interval. This study relies on the results obtained in the NS2-based simulator, but the authors do not use another tool to validate their results. Additionally, the performance is based only evaluating only ND message interval, as we can see in our experiments, performance is also impacted by different values of the ND message hold time.

Chapter 3: Experimental Plan

In this chapter we will explain the experimental methodology applied in the physical testbed and NS3-based simulator used to validate the results. The objective to construct a physical testbed capable to simulate topology changes over time without the need to physically move the devices; the design of the testbed also allows to conduct different levels of mobility by controlling the number of links that change in the topology transitions, and the topology transition time. Our the testbed must be capable to uncover the relationship between ND message interval and ND message hold time that minimizes the overall packet loss rate. We begin describing the hardware used, the characteristics of the wireless transceivers used in the wireless transmission, and the characteristics of the MAC protocol used in the standard of these transceivers. We will continue with a description of the iptables, the Linux firewall used to simulate the topology changes. Next we describe the OLSR daemon used for routing protocol; then, we introduce the GNU ping program and explain how this program is used to generate *data traffic*, the last component used is the HDF5 a hierarchical database used to collect the statistics of the data. We will conclude this section describing how each component is integrated to run the experiments. In the next section, we describe the NS3-based simulator's models used to validate the observations and results obtained in the physical testbed. We conclude this chapter with the experimental plan.

3.1 Physical Testbed

The physical testbed is composed of 6 nodes, 1 master and 5 nodes. The purpose of the master node is to start the experiment in all nodes, and to ensure that all nodes will synchronize the topology changes during the length of the experiments.

Master server configuration

- *CPU*: Intel Core 2 Duo @ 2.8GHz
- *RAM*: 4GB
- *Storage*: 150GB hard drive

Node 1, node 2, and node 3 configuration

- *CPU*: Intel Core 2 Duo @ 2.8GHz
- *RAM*: 4GB
- *Storage*: 250GB hard drive

Nodes 4 and node 5 (OMAP4430 pandaboard rev A4).

- *CPU*: Armv7 Processor rev. 3
- *RAM*: 1GB
- *Storage*: 32GB SD card

All nodes use Ubuntu version 12.04. This version was chosen because it is compatible with the version of pandaboards used. All nodes are within transmission range and the antennas of the wireless cards used are located 24 inches between each other.

For wireless communication, all nodes use a TP-Link TL-WN722N, a USB external transceiver with following communications characteristics:

Wireless Standards	IEEE 802.11n, IEEE 802.11g, IEEE 802.11b
Wireless Security Support	64/128 bit WEP, WPA-PSK/WPA2-PSK
Frequency	2.400 - 2.4835GHz
Modulation Technology	DBPSK, DQPSK, CCK, OFDM, 16-QAM, 64-QAM
Signal Rate 11n:	Up to 150Mbps(dynamic)
11g :	Up to 54Mbps(dynamic)
11b :	Up to 11Mbps(dynamic)
Reception Sensitivity130M:	-68dBm@10% PER
	108M: -68dBm@10% PER
	54M: -68dBm@10% PER
	11M: -85dBm@8% PER
	6M: -88dBm@10% PER
	1M: -90dBm@8% PER
Transmit Power	$<$ 20dBm
Wireless Modes	Ad-Hoc / Infrastructure mode

802.11 MAC Protocol

As mentioned in Section 1.2.1, WiFi IEEE 802.11 uses CSMA/CA (carrier sense multiple access with collision avoidance), a random access protocol in which each node senses the channel before it starts transmitting and it will refrain transmissions when the channel is sensed busy to avoid collisions. Once a node begins to transmit a frame, it transmits the entire frame and once the frame is completely received from the final destination, the destination waits a short period of time called *Short Inter-frame Spacing* (SIFS). As a result, it sends an acknowledgment. If the acknowledgment is not received after a fixed number of re-transmissions, the frame is discarded. When a node has to transmit, it will follow these steps:

1. If the transmitting node senses that the channel is idle, it will start the transmission of its frame and waits for a short period of time; this time is called *distributed inter-frame space*

(DIFS).

2. Otherwise, the station will choose a random backoff value. This value is a count-down to sense the channel again. If the channel remains busy, the counter value remains in waiting status.

3. When the counter reaches zero, the station transmits the entire frame; after the destination receives the frame, it waits for a *short inter-frame space* (SIFS) and sends an acknowledgment *ACK* package back to the sender.

4. If the acknowledgement is received, the transmitting node knows that its frame has been correctly received at the destination node. If the transmitting node has another frame to send, it will start at step 2. Otherwise when the acknowledgment is not received, the transmitting node reenters the backoff phase in step 2 with the random value chosen, but this time waits for a larger interval of time [3].

Routing Protocol

The testbed uses OLSRd rev 0.6.3, OLSRd is a daemon based on the OLSR protocol with two important features: it requires low CPU processing to save power consumption, and handles a high level of scalability (up to 2000 nodes)[24].

OLSR (*Optimized Link State Routing)* is a proactive protocol, based in a hop-to-hop scheme that inherits the stability of *link state* protocols and optimizes the performance by implementing *Multipoint Relays*, reducing the re-transmission of control packets, and leading to an efficient flooding of control messages in the network [9].

OLSR does not require reliable transmission of control messages. Each node sends control messages periodically and can sustain a reasonable loss of control packets due to interference in the radio transmission channel [1]. This protocol is suitable for large and dense networks due to only MPR nodes disseminate *topology control* information in the MANET, the improvement is more visible as the number of nodes increases in the network [25]. A problem in this scheme is when the distance between nodes increases that limits the number of MPRs. The worst case is

when all nodes in the network (except nodes at the edge of the network) become MPRs and the network becomes a pure link state protocol.

The core functionality of OLSR are the links and neighbors discovery process, topology discovery process, and routing table calculation process.

Link Sensing and Neighbor Discovery

Link Sensing

In MANETs, the communication between one-hop neighbors is done by discovering direct paths or links by broadcasting ND messages called *hello messages* in OLSR. A link is a pair of interfaces (from two different nodes) susceptible to hearing one another (i.e. one may be able to receive the traffic from the other). A node is said to have a link to another node when one of its interfaces directly communicates to one interface of another node [1]. As explained in Section 1.2.1, decreasing signal strength, interference from other sources, and multipath propagation are some of the major challenges in wireless links [3]. OLSR protocol uses *link sensing* to detect one-hop neighbors. In OLSR, all nodes maintain a *link set*, the set of links with one-hop neighbors.

A *link set* is composed of the local interface address (L_local_iface_addr), one-hop neighbor interface address (L_neighbor_iface_addr), the time the link between the two nodes is symmetric (L_SYM_time), the time can hear the one-hop neighbor (L_ASYM_time), and the time when the link expires (L_time); when a link expires, it *MUST* be considered *lost*.

There are 4 types of links:

- *UNSPEC_LINK* where no specific information of the link is given
- *ASYM_LINK* that indicates the link is asymmetric, or the neighbor is âĂŸheardâĂŹ
- *SYM_LINK* indicates that the link is symmetric with the interface
- *LOST_LINK* indicates that the link is no longer announced

L_SYM_time defines the link type, if L_SYM_time is not expired, the link must be declared as symmetric, but if the L_SYM_time expires, the link *MUST* be declared as asymmetric until

L_ASYM_time expires, then the state of the link *MUST* be declared as lost [1].

Neighbor Discovery

Neighbor Discovery is based on the periodically dissemination of *hello messages* to discover one-hop neighbors and maintain its one-hop neighbor set information for a pre-defined period of time. Nodes recognize their one-hop neighbors by classifying links status. Along with the link information, neighbor status is also announced. There are 3 types of neighbors status:

- *SYM_NEIGH* indicates that the neighbor have at least one symmetrical link with this node
- *MPR_NEIGH* indicates that the link has at least one symmetrical link and has been selected as MPR by the sender
- *NOT_NEIGH* when the neighbor is no longer a neighbor or is not a symmetric neighbor

In the ND process, each node must detect its one-hop neighbor nodes with which it is connected with a *symmetric link*, different factors in the transmission channel like collision, interference, noise can make some links become uni-directional for short periods of time. Therefore all links must be validated in both directions to be considered valid. To accomplish this, each node must disseminate *hello messages* in a certain period of time to continually maintain up-to-date information about its links and neighbor's status.

Hello Messages

. *Hello messages* are the mechanism to discover links and one-hop neighbors. Figure. 3.1 shows the format of a *hello message* in OLSR.

Hello messages are part of the data portion of the general packet format that will be explained later. When a packet includes a *hello message*, the message type set is HELLO_MESSAGE, the time to live (TTL) of the message is set to 1 and *vtime* is set according to the value of NEIGHBOR_HOLD_TIME or *hello validity*.

Next we explain the fields of the *hello message* as defined in the RFC 3626.

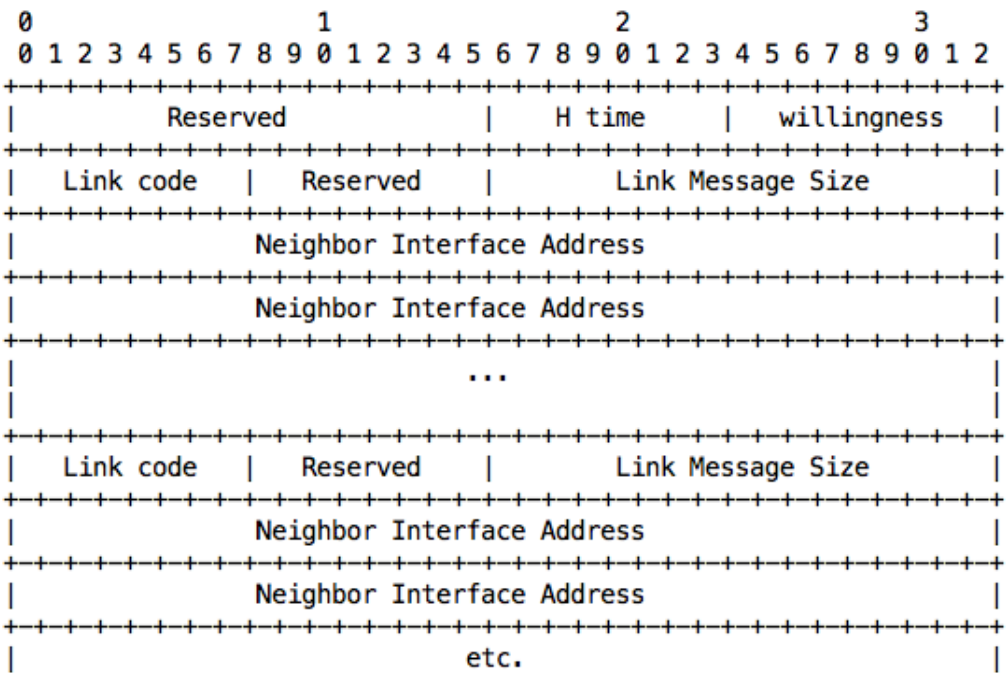


Figure 3.1: Layout of OLSR *hello message* format.

Reserved field must set to "000000000000000".

H time represents the time before the transmission of the next *hello message*, the hello emission interval is represented by the mantissa (four highest bits of the Htime field) and by its exponent (four lowest bits of Htime field).

Hello emission interval=C*(1+a/16)*2∧b) [in seconds].

Where a is the integer integer represented by the four highest bits of the Htime field and b is the integer represented by the four lowest bits of the Htime field.

Willingness This is the willingness for a node to become MPR and forward data for one hop neighbors. There are five different values of willingness, 0 WILL_NEVER, 1 WILL_LOW, 3 WILL_DEFAULT, 6 WILL_HIGH, and 7 WILL_ALWAYS, all nodes start with WILL_DEFAULT and according to power capabilities will change, for low power to WILL_LOW or WILL_NEVER, for nodes with high power capabilities WILL_HIGH or WILL_ALWAYS, and change as power capabilities changes.

Link code In this field OLSR provides information about the link state and the neighbor state;

invalid codes are silently discarded by the nodes.

Link Message Size Includes the link code field and all Neighbor Interface addresses.

Neighbor Interface Address An address of the interface of a neighbor node.

Hello messages serve for 3 purposes:

- Link sensing
- Neighbor detection
- MPR selection signal

Hello message's information is generated base on the local link set, neighbor set, and MPR set of each node. The links and neighbor status are managed with following algorithm:

Link type set

if L_SYM_time $\geq$ current time (not expired).

then Link type = SYM_LINK

else if L_ASYM_time $\geq$ current time (not expired) AND L_SYM_time $<$ current time (not expired)

Link type = ASYM_LINK

else if L_ASYM_time $<$ current time (expired) AND L_SYM_time $<$ current time (expired)

Link type = LOST_LINK

Neighbor type set

if the main address, corresponding to L_neighbor_iface_addr, is included in the MPR set:

Neighbor type = MPR_NEIGH

otherwise , if the main address, corresponding to L_neighbor_iface_addr is included in the neighbor set:

if N_status == SYM

Neighbor type = SYM_NEIGH

otherwise , if N_status == NOT_SYM

Neighbor type = NOT_NEIGH

Hello messages permit each node to learn the neighbor set of its neighbors up to *two-hop neighbors*; based on this information, each node performs the selection of MPRs and updates its routing table with the information of the status of its links and one-hop neighbors.

Topology Discovery

The topology control *TC messages* information allows to all nodes have a complete view of the network. Since TC message contain the intra-forwarding information required for routing purposes, each node maintained this information in the routing table. *TC messages* are broadcast within an interval of time. Only *MPR* nodes are in charge to generate and periodically forward *TC* messages. *TC messages* are forwarded when *TC validity* time expires and when *MPR* nodes detect changes in its *MPR selector set* [1], when the *MPR selector set* settle, *TC messages* disseminate on a regular basis; this scheme guarantees that all nodes will keep up-to-date topology information.

Topology set is the topology information that each node maintains of the network, and is obtained from the *TC messages*; *Topology set* information is used in the *routing tables* to calculate the paths when the node needs to send a packet to a neighbor. Each node maintains following information: the main address of a destination node (T_dest_addr), the address of the MPR of the destination node (T_last_addr), the sequence number (T_seq), and the time at which this tuple expires and *MUST* be removed.

Topology Tuple (T_dest_addr, T_last_addr, T_seq, T_time).

TC Message Format

Next we explain the fields of the *TC message format* as defined in the RFC 3626.

Advertised Neighbor Sequence Number (ANSN) A sequence number is associated with the advertised neighbor set. Every time a node detects a change in the advertised neighbor set, it increments the sequence number. This number is sent in this ANSI field of the *TC message* to keep track of the most recent information. When a node receives a *TC message*, it can decide on the basis of this Advertised Neighbor Sequence Number.

Reserved field must set to "0000000000000000".

Advertised Neighbor Main Address This field contains the main address of a neighbor node and all neighbor addresses of the advertised neighbors of the Originator node are put in the *TC message*. If the maximum allowed message size is reached, additional *TC messages* are sent until the entire advertised neighbor set has been sent.

TC messages are sent in the data-portion of the general message format with the "Message Type" set to TC_MESSAGE, the TTL is set to 255 (maximum value) to diffuse the message into the entire network, and *vtime* set accordingly to the value of TOP_HOLD_TIME or *TC message validity time*, the format of the *TC message* is show in fig 3.2.

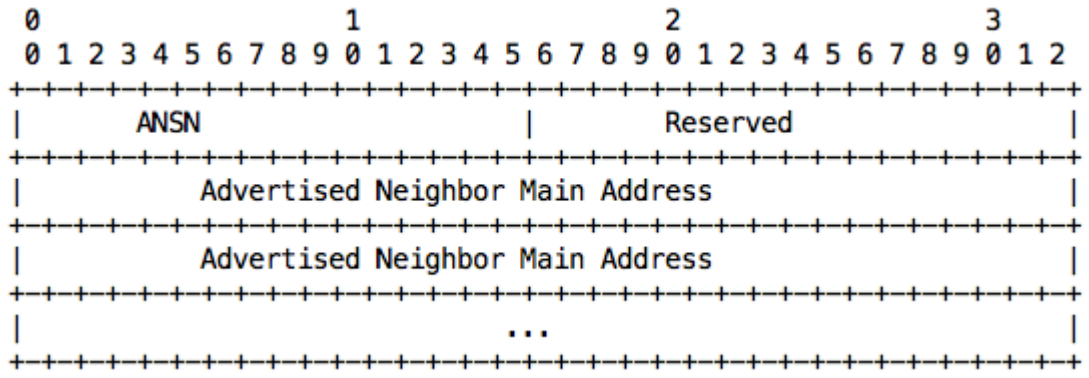


Figure 3.2: Layout of TC message.

MPR Neighbors

MPR neighbors are one-hop neighbors in charge of disseminate topology information by broadcasting *TC* messages and to forward data packets to two-hop neighbors.

The purpose of MPR is to minimize the flooding of broadcast control messages in the network and reduce duplicate re-transmissions in the same region. Each node in the network selects a set of nodes in its neighbor set that re-transmits its data packets. This set of selected nodes is the *MPR set* of that node.

As shown in Figure 3.3, one-hop neighbors of node m, can read and process packets sent by node m, but will not re-transmit packets send by node m. This node selects its *MPR set* among its one-hop neighbors such that the set of MPR selected covers all the nodes that are two-hop neighbors [25]. The *MPR set* of node m, or MPR(m), is an arbitrary subset of the neighbor set of node m that satisfies the following conditions: every two-hop neighbors of node m that are connected to a candidate of MPR *MUST* have a *symmetric link* with the MPR. A *MPR selector* is a group of nodes that selected a node to be their *MPR*.

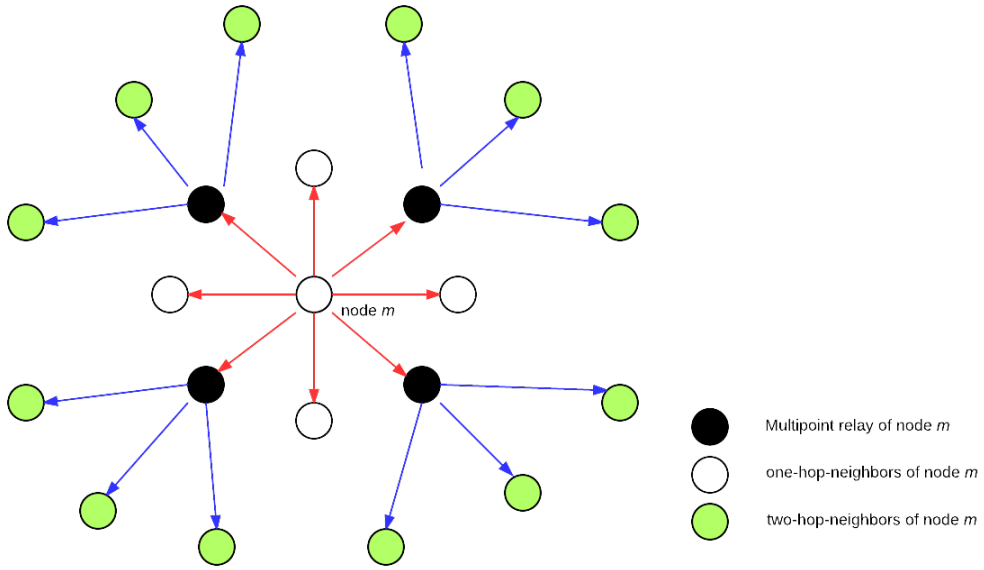


Figure 3.3: OLSR Multipoint relay.

Routing Table Calculation

A *routing table* is a database that each node maintains with the necessary information to route packets to all nodes in the MANET. When a node receives a TC message, it will parse the

message and store the information in the form of the *Topology Tuple* in the routing table. Routes are calculated from this database by tracking the connected pairs in a descending order.

As Figure 3.4 shows, to route a packet from node m to node n, node m keeps in its *routing table* the destination address of node n (the address provided by the last TC message from its *MPR* neighbor). The route information begins with the tuple (X, n), that is the address of the last-hop node to the destination (*MPR (n)*); the sequence number of nodes X to route the packet to node n; and the validity time associated to this entry in the *routing table* of node m, when the entry expires, it is removed from the *routing table*. As shown in 3.4, to find the shortest path from node m to remote node n, node m needs to find a connected tuple (X,n), then a connected pair (Y,X), and so forth until node n finds a pair (MPR,Y). The sequence number in the "Topology Tuple" is used to detect connected pairs which have been validated by further topology changes.

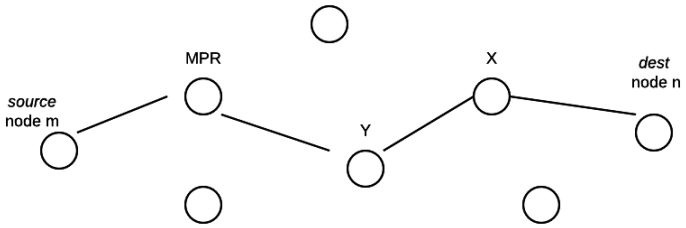


Figure 3.4: Building a routing from routing table.

The routing table information is also generated from hello messages; therefore, if any information coming from a hello message or TC message is different from current *routing table* data, the routing table is re-calculated. The *routing table* is recalculated only when a change in the neighbor set is detected concerning a *symmetric link* or when a route to any destination has expired. The *routing table* is updated when a change is detected in:

- Link set
- Neighbor set
- The 2-hop neighbor set

- The topology set
- The Multiple Interface Association Information Base

To recalculate the routing table, nodes follow the bellow steps:

1. All entries in the table are removed.
2. The new entries are recorded in the table, starting with the one-hop neighbors (*d=1*) as destination nodes. For each neighbor entry in the table, whose link status is symmetric, a new route entry is recorded in the *routing table*, where destination and next-hop address fields are both set to the address of the one-hop neighbor, and the distance is set to 1.
3. The next route entries to be updated in the *routing table* are for destination nodes *d+1* hops away, then increments to nodes *d+2*, and so forth.

OLSR Packet format

Packets in OLSR routing protocol follow a standard configuration to maintain backward compatibility and different types of messages can be *piggybacking* in one packet. OLSR packets have a standard information, as shown in Figure 3.5.

Packet Length the length in bytes of the packet.

Packet Sequence Number is a unique value assigned to each interface of each node, and each packet transmitted has a unique packet sequence number. When a packet length is less than the packet header size (16 bytes), OLSR silently discards the packet.

Message Header

Message Type OLSR classifies different types of messages (0-127), in this field the type is specified.

Vtime is the validity time at the reception of the packet; the validity time of the packets is calculated based on following criteria:

Validity time=C*(1+a/16)*2∧b) [in seconds].

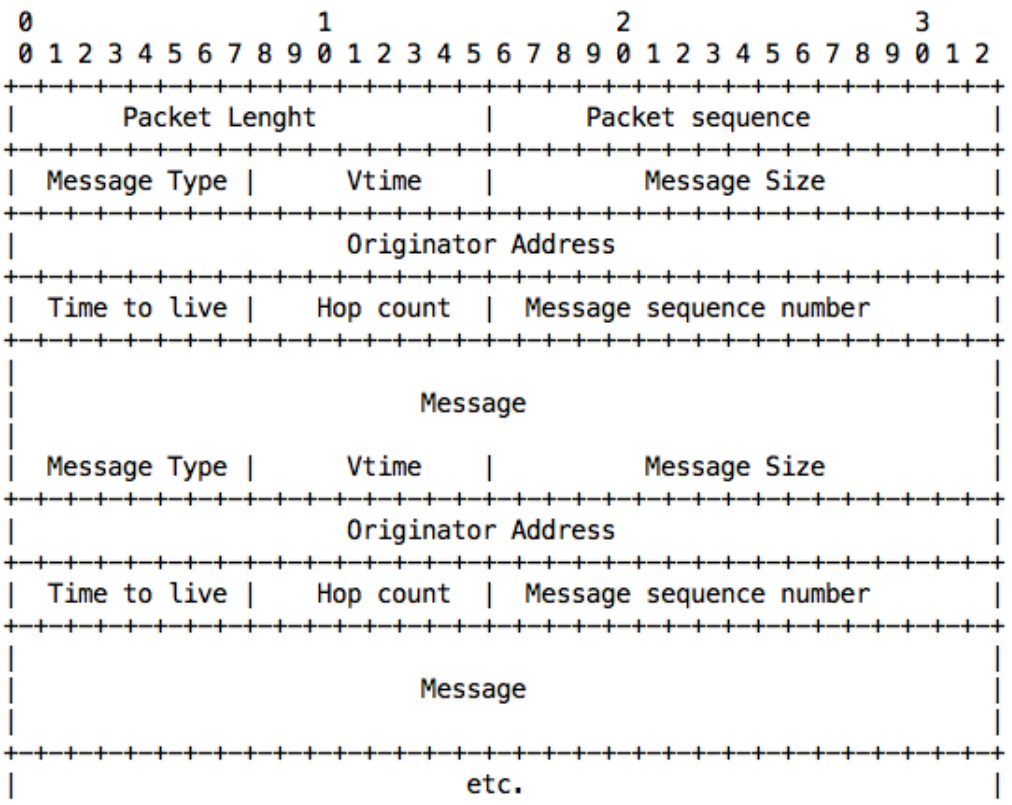


Figure 3.5: Layout of OLSR general packet format.

Where a is the integer represented by the four highest bits of the *vtime* field and b is the integer represented by the four lowest bits of the *vtime* field. The value of C is specified in terms of the type of message (*Hello* message, *TC* message, or *HNA* message).

Message Size includes all information of the message, including the *message header* information.

Originator address is the address of the node that generates the message.

Time To Live is the number of hops that a packet can be re-transmitted, if this number is 0 or 1 and the packet is not in the final destination, the packet *MUST* be discarded.

Hop Count This field start with 0 when the message is generated, and the number is incremented by 1 when the message is re-transmitted.

Message Sequence Number Each message that is generated will have a unique starting number and consecutive messages will be assigned a sequential number of the original incre-

mented by one.

Iptables

Linux kernel includes a subsystem to process network packets and this subsystem is called Netfilter or firewall; iptables is a powerful tool available in Linux kernel 2.4.x and above to manage the Netfilter. Iptables is configured in the kernel and only superusers accounts can manage it. It is also considered an inter-layer tool because it can manipulate packets in the IP layer and packets in the transport layer (e.g., TCP, UDP, ICMP). Iptables is composed of 4 tables (RAW, NAT, MANGLE, and FILTER), these tables are controlled with chains and *rules* applied to data packets when these are sent, received or, forwarded. In our experiments, the rules are applied to the iptables only when packets are received; this is explained in more detail later in this section. Another manager for Netfilter is ipchain; this is a stateless packet filter that will accept, drop or forward packets regardless of the sender. Iptables is a stateful packet filter or IP filtering that applies several different actions to packets that traverse it based on the sender's address, receiver's address or both; bellow is a list of the key terms that define iptables *chains* and *rules*:

Drop/Deny When a packet traverses the IP filtering of the firewall and this drops the packet, the packet simply disappears, and no further notification is sent. Sender is not awarded of the packet status in any way.

Reject A packet that is rejected differs from the dropped that there is a notification to the sender of the packet either if the sender is another hop or is an application program in the firewall.

State All packets that traverse the iptables are assigned a state, for new packets in a stream will be assigned state *new* to a packets that has first arrived, or *established* if the packet is part of a stream of previous packets already processed by the iptables.

Accept A firewall accepts a packet when it complies with the set of rules established in the iptables.

Match A single match that tells a rule that the packet header must contain specific information, or the whole packet must fulfill specific information.

Rule is a set of matches that applies to a single target or a set of targets or actions per rule.

Ruleset The complete set of rules that apply for *raw*, *nat*, *filter*, and *mangle* tables.

Target There is a *target set* of each rule in a ruleset. The target specifies what actions follow with the packet (e.g. accept it, drop/deny it, or reject it).

Jump A jump instruction is written similar to a target, with the exception that instead of write the name of another chain.

Policy Two types of policies apply for iptables, first, the ones applies in chains called chain policies, these apply for one or more IP addresses, second, the security policies, that apply to the whole network.

Chain is a set of rules applied. Each table has certain number of rules; therefore, chains are applied to the packets when these traverse certain table (e.g., a chain of rules can apply only for packets from a source directed only to the destination or only to forwarding packets, or both).

Tables Iptables is composed of four tables each with a specific function:

- *RAW* Is the first table used to filter packets. It is mainly used for exceptions; it applies for PREROUTING or packets arriving via network, or OUTPUT, packets locally-generated saving memory-demanding operations such as connection tracking.
- *NAT* Apply for Network Address Translation function; this table is created when a packet creates a new connection, consists of PREROUTING (altering packets as soon as they come in), OUTPUT (altering packets locally generated before leave out), and POSTROUTING (altering packets not generated in the firewall before they leave out).
- *MANGLE* Used for specialized packet alteration; it includes PREROUTING that alters packets before they are routing, either for the firewall or packets that the firewall

attempt to forward or both, OUTPUT apply for packets locally-generated and are ready to leave-out, INPUT apply for packets intended only for the firewall.

- *FILTER* Is the default table. It applies for INPUT packets intended to the firewall, FORWARD packets being routed by the firewall, and not generated locally and OUTPUT packets not generated locally.

Ping

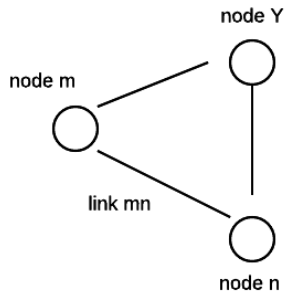
Ping is a tool to monitor the response of the network. It is considered a system administrators' tool. It is based on ICMP Internet control message protocol and consists of a small 64 bytes packet with 8 bytes in the header and 56 bytes of data. The node that sends the packet waits for a return confirmation packet (RTT). If target node is reachable and available, a good return packet will be received; ping provides useful information to know the status of the network, like how many hops exist between the sender and receiver, and the amount of time takes a packet to be received. Additionally, ping is used to measure the network performance since it can notify the number of packets are loss during a period of transmission [26].

HDF5

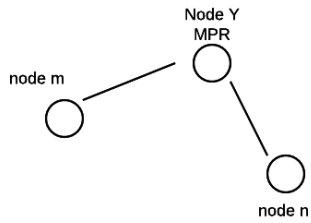
Hierarchical Data Format version 5 (HDF5) is an open source library and file format with high capacity to store a large amount of numerical data. Originally developed at NCSA. It is a *defacto standard technology* in Python to store large numerical datasets. Designed primarily for scientific usage, HDF5 is a mechanism to store large homogeneous numerical type arrays. The data model is organized into: *groups*; hierarchical containers that store datasets and other groups; groups are arranged similar to a filesystems that contains files and folders; *datasets* array-like objects that store numerical data on disk; *attributes*, are user-defined bits of metadata that can be attached to datasets and groups [27]. HDF5 handles endianess, therefore, files can be accessed and manipulated in different filesystems. HDF5 was installed in each node and used to collect the ping results to each neighbor.

Topologies

A total of 36 different topologies were utilized in our experiments; these topologies were emulated using iptables *rules*. In this section we explain the emulation of the topologies used and topologies transitions used to model mobility. Mobility is one of the most difficult features to model in physical experiments in MANETs due to spatial dependencies, geographical restrictions, and temporal dependencies [28]. To explain the construction of the topologies followed, in Figure 3.6a we illustrate a simple network of 3 nodes that can maintain direct communication, when node m transmits data packets to node n, node m can identify two routes: send packets through *link mn*, or send packets to node Y, and node Y forwards the packets to node n (if node Y is selected as MPR of nodes m and n); node m forwards the packets through *link mn* to communicate with node n; following the shortest path. If node n moves from the transmission range of node n and remains in transmission range with node Y, *link mn* no longer exists, to continue the communication, the packets need to be routed through node Y, as seen in Figure 3.6b. Node n will reply back to node m following the same route through node Y.



(a) Nodes using link(mn) for communication



(b) Nodes mn using node Y for communication

Figure 3.6: Node m and n communication schemes

To simulate aforementioned mobility in node n, following configuration in the iptables is completed in node n:

sudo iptables -t raw -A PREROUTING -m mac –mac-source node m -j DROP

Similar configuration in iptables needs to be done in node m as follows:

sudo iptables -t raw -A PREROUTING -m mac –mac-source node n -j DROP

According to above configurations, raw tables are configured to drop the arriving packets from the two nodes in the PREROUTING stage (i.e., before packets are processed). The two nodes need to apply the rules to the iptables at the same time to accomplish the effect of mobility. When node m transmits packets to node C, the *raw* table in node n will drop all arriving packets and will not notify node n. In this moment, OLSR protocol in node m recalculates the route to send packets to node n; if node m's routing table finds node n as two-hop neighbor, node Y will route packets between these two nodes and will re-route all packets for node n through node Y, when node n receives the packets from node Y, the firewall will process them. If this rule is removed from the iptables in node m and n, *link mn* will be re-established, and the routing tables will be recalculated, then *link mn* will be used as the shortest path.

3.1.1 Experimental Description

After mentioning the core elements that govern our experiments, we will proceed to list the steps followed to run our experiments.

The testing environment is started with a python script running in each node; the purpose of the python script is to coordinate the flow of events and execution of the programs during the experiments.

As previously explained, all nodes must be synchronized to start the experiment, due to each node is an individual system. One node is the master that starts a python script and this script runs a ssh command. Ssh is a UNIX-based command interface and protocol to gain access to the rest of the nodes and starts the python script.

Since all nodes are located within transmission range, all of them have a bi-directional communication with each other. To limit the communication, the python script calls iptables to modify the *RAW* tables and *DROPs* all packets received from nodes we want to cancel the bi-directional communication. This is done configuring the *RAW* tables to *DROP* packages from

connected links and creates the first topology. The construction of the rest topologies in the experiment, follows the same procedure.

Olsrd daemon is initialized in each node, with the first ND parameters tuples (δ,τ) values as indicated in Table 3.1. Olsrd requires few seconds to complete its configuration.

During the experiments, the GNU ping program is used to generate data traffic, each node sends ping messages to all nodes in the MANET. To avoid collision, a random amount of time is chosen by each node to begin its data transmission and avoid synchronization with other nodes. When a tuple of nodes n and m begin data transmission, OLSR protocol will choose the shortest path between them (*link mn*), rather than use node Y (see Figure 3.6a); if *link mn* is broken or expires, for the remaining packets, the two nodes need to find an MPR to re-route the packets. As Figure 3.6b illustrates, node Y is selected as MPR(n) and MPR(m), all packets are re-routed through it.

All experiments were run with a *topology change interval* (TCI) of 120 seconds. When the TCI expires, the iptables are reset and adjusted according to the next topology; all nodes synchronize iptables changes every TCI to maintain valid topologies. During the experiment 9 different topologies were cycled until 160 topology changes occurred.

3.2 NS3-based Simulator

In order to validate the results obtained from the physical testbed, a NS3-based simulator was built. NS3 is an open source discrete-event network simulator developed for internet systems, targeted primarily for research and educational purposes [29]. The NS3-based simulator follows the same physical testbed architecture to build the simulator; we have used several build-in models to create the physical layer, link layer, data layer, internet layer, transport layer, and application layer in each node.

ConstantPositionMobility is used to model the physical layer. This model allows to simulate discrete movements in the simulator during the execution of the experiment, changing a nodeâĂŹs from a location to a different location in a specific period of time. For our experiments, the period

of time chosen is the TCI=120 seconds, when the network simulates a topology change.

YansWifiPhy is a model used to simulate 802.11a WiFi transceiver instance in each NS3 node. We make use of the Helper class WiFiHelper, a specialization inside the specialization WiFiNetDevice located inside NetDevice. This model relies in the PropagationDelay and PropagationLoss models that are part of the YansWiFichannel model.

The PropagationDelay model used is the constant speed PropagationDelay model, which is defined as a constant delay of two-thirds the speed of the light. A propagation loss can be related to different root causes; one attribute defined by the YansWiFiPhy model is the energy detection threshold. Where the energy of the arriving signal should be higher than this threshold, that is a propagation loss model. The range of propagation loss model allows the user to define a threshold of transmission distance. If a neighbor's distance is less or equal than the threshold, the device in the physical platform will receive the wireless at full strength. On the other hand, if the transmission distance, is larger than the threshold, the signal is not processed. YansWiFiChannel model is configured with the PropagationDelay and PropagationLoss models; it is set as the channel for the YansWiFiPhy model.

NS3 includes AdHoc WiFiMac model, the corresponding WiFiMachelper for ad-hoc simulation. This model was used to simulate the MANET in the NS3-based simulator.

The routing protocol used in the simulation is the OlsrHelper, a class that configures the NS-3 OLSR model.

InternetStackhelper is composed of the data link, network, and transport layer modeling; for the data link, NS-3 models the Address Resolution Protocol (ARP); for the network layer, the Internet Protocol is simulated; and, for Transport layer, are simulated the TCP/UDP protocols.

The V4ping model is used to generate data traffic; this model sends an ICMP packet from each node to the other five nodes every second. The time to start the ping transmission is chosen randomly to avoid transmission synchronizations with the other nodes and minimize collisions.

Finally, the NS3-based simulator monitors the neighbor set of each node, and records in a file the symmetric neighbor's information.

3.3 Experimental Plan

In Section 3.1.1, we described the experimental procedure and the length of the experiments using different tuples of (δ,τ), a total of 62 different experiments were completed with different tuples.

We are interested in finding the ND discovery parameters configuration that ensures the optimal performance in MANETs with considerable mobility. To accomplish this goal, in Section 1.1, we analyze the impact of the ND parameters in the reliability of the ND discovery process; therefore, we explore different settings of the ND parameters running different sets of experiments, varying the ND message interval (δ) values of 2 sec, 4 sec, 8 sec, 16 sec and 32 sec, and for each ND message interval, varying a maximum of 16 different ND message hold time. We have used a workload characterization of *constant bit rate* (CBR) of the GNU ping, where each node in the MANET is a generator and receiver from all other neighbors. CBR models is a good approach for testing the stability and the congestion reaction of our testbed. Table 3.1 summarizes the different *ND parameters* chosen in our experiments.

Table 3.1: Values of ND message interval and ND message hold time with TCI= 120 seconds.

τ (sec)	δ = 2 sec	δ = 4 sec	δ = 8 sec	δ = 16 sec	δ = 32 sec
(1.00δ)	2.00	4.00	8.00	16.00	32.00
(1.10δ)	2.20	4.40	8.80	17.60	35.20
(1.25δ)	2.50	5.00	10.00	20.00	40.00
(1.50δ)	3.00	6.00	12.00	24.00	48.00
(2.00δ)	4.00	8.00	16.00	32.00	64.00
(2.25δ)	4.50	9.00	18.00	36.00	72.00
(2.50δ)	5.00	10.00	20.00	40.00	80.00
(3.00δ)	6.00	12.00	24.00	48.00	96.00
(3.25δ)	6.50	13.00	26.00	52.00	104.00
(3.50δ)	7.00	14.00	28.00	56.00	NA
(4.00δ)	8.00	16.00	32.00	64.00	NA
(6.00δ)	12.00	24.00	48.00	96.00	NA
(8.00δ)	16.00	32.00	64.00	NA	NA
(16.00δ)	32.00	64.00	NA	NA	NA

As can be seen in Table 3.1, some tuples (δ, τ) were marked as invalid (NA); the reason is

that the *ND message hold time* is larger than the TCI. When the TCI expires and the topology has changed, some neighbors may be out of transmission range and the node still considers these neighbors as valid. All packets send to these neighbors are lost until the node recognizes that these neighbors are gone. This approach generates high packet loss rates.

Chapter 4: Results

In this chapter, we present the results obtained from our experiments. In the first section we present the overall packet loss rate obtained from our experiments in the NS3-based simulator and the physical testbed. In the next section, we expose our observations of the results obtained in each experiment and explain the existence of neighbor-flapping, an unstable state phenomenon derived from the ND parameters. We explain how it is generated, and how it is eliminated. We conclude this chapter with the analysis of our observations.

4.1 Experimental Data

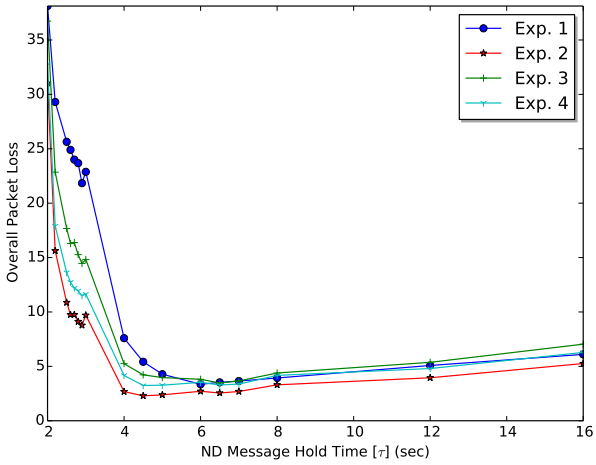
In MANETs different metrics can be used to measure the performance, based on *user performance*, *resource utilization*, and *systems metrics* [2]. To measure the optimal performance of the routing protocols, we can measure: end-to-end throughput, average link utilization, average packet loss probability, energy efficiency, protocol overhead, packet loss, etc. As mentioned in section 3.3, the experiments run a CBR data traffic with GNU ping packets. In our experiments we measured the *overall packet loss rate*.

4.1.1 Overall Packet Loss Rate

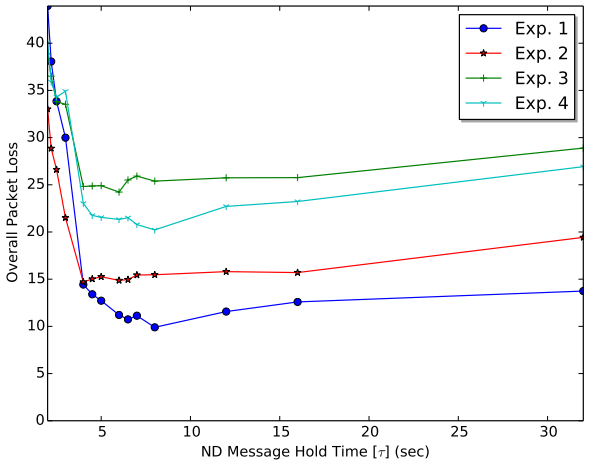
The overall packet loss rate is the total number of packets transmitted minus the total number of packets confirmed by each receiver to its sender. The result is the total number of packet loss during the transmission. This result is divided by the total number of packets transmitted. The overall packet loss rate is a metric that measures the stability and reliability of the one-hop links and one-hop neighbors information defined by the ND process parameters.

In the Plots 4.1 and 4.2, we present the results of the 4 sets of experiments, each experiment ran 9 different topologies following a TCI of 120 sec; in experiment 1, topologies change one link, this simulates a low-mobility MANET. In experiments 2, 3, and 4, topologies follow a random number of links change to simulate high mobility. In the left side of the plots, we can see the overall packet loss rate results obtained from the NS3-based simulator. On the right side, we

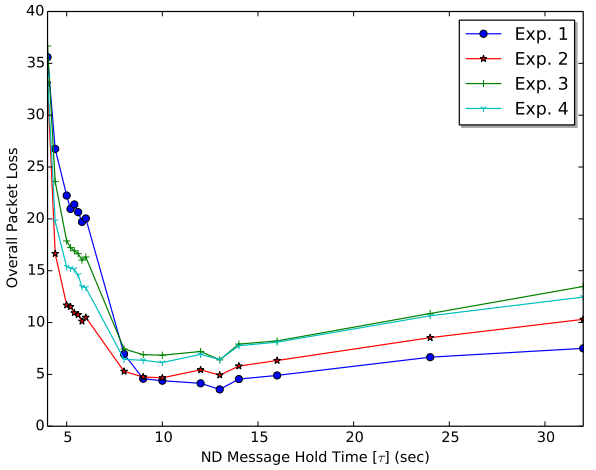
present the overall packet loss obtained from the physical experiments.



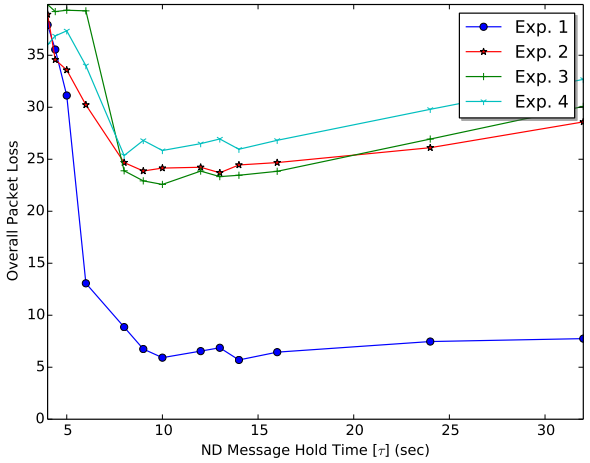
a) (δ) = 2 sec NS3-based simulation



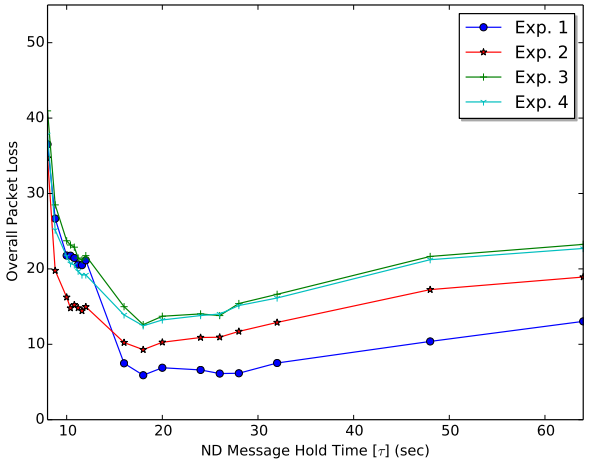
b) (δ) = 2 sec physical experiments



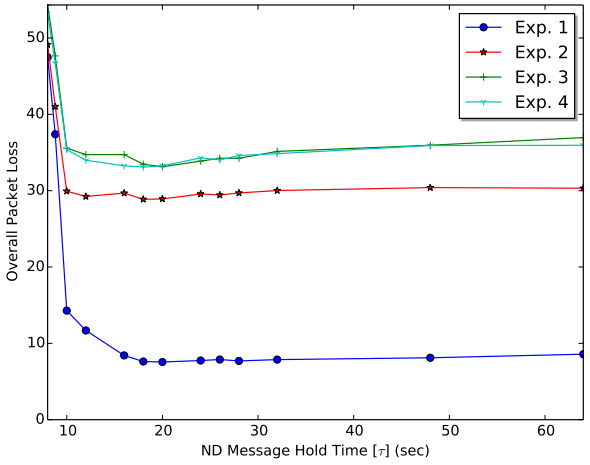
c) (δ) = 4 sec NS3-based simulation



d) (δ) = 4 sec physical experiments

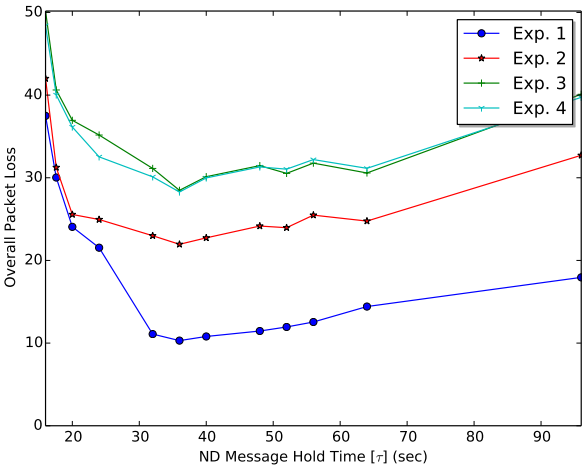


e) (δ) = 8 sec NS3-based simulation

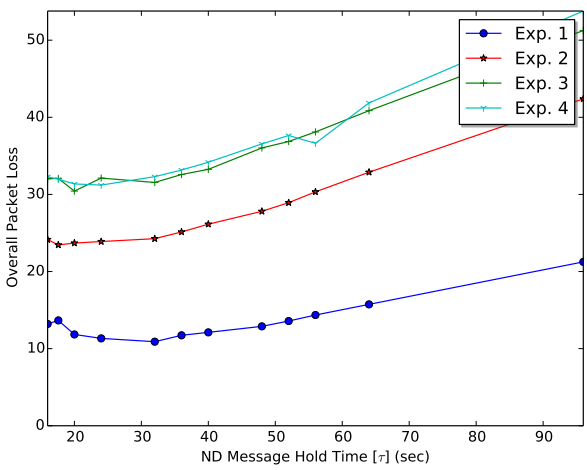


f) (δ) = 8 sec physical experiments

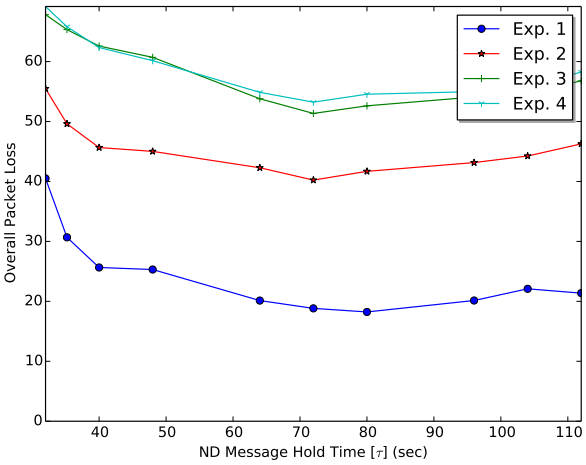
Figure 4.1: Overall packet loss rate



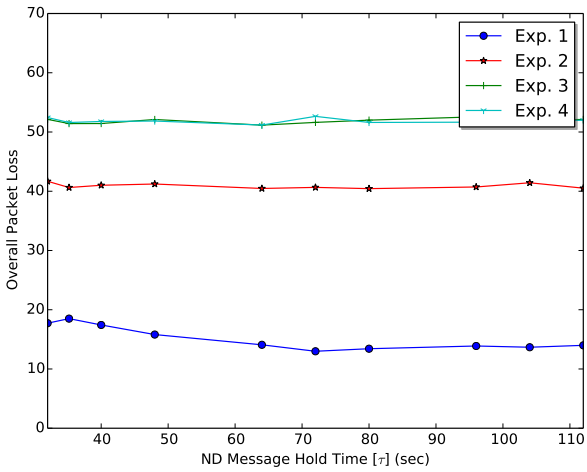
a) (δ) = 16 sec NS3-based simulation



b) (δ) = 16 sec physical experiments



c) (δ) = 32 sec NS3-based simulation



d) (δ) = 32 sec physical experiments

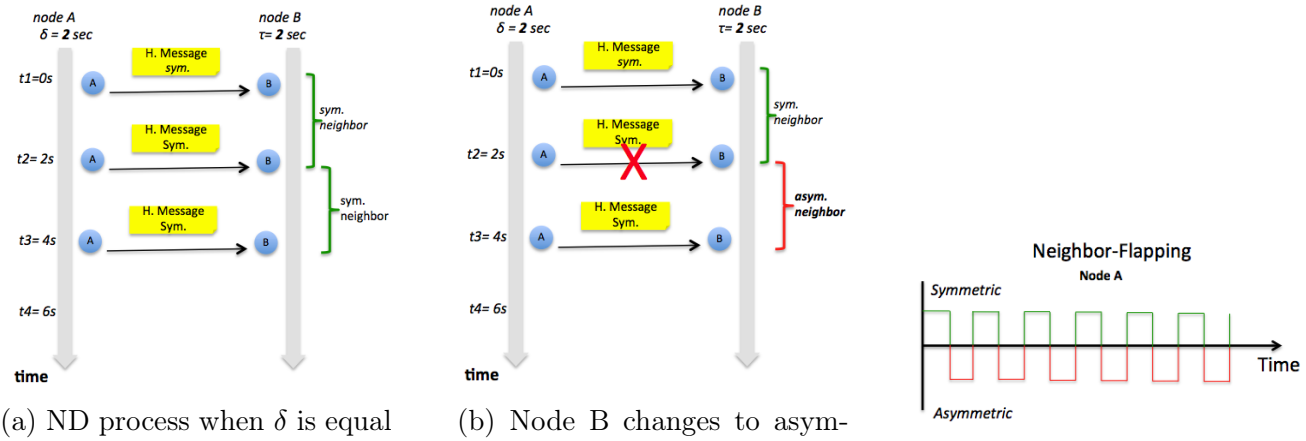
Figure 4.2: Overall packet loss rate 2

4.2 Observations

Observation 1: According to the results in plots 4.1 and 4.2, we can observe the highest overall packet loss rate when the ND message hold time is equal to the ND message interval; in this scenario, ND messages are expected to be received when the ND message hold time expires, in an ideal transmission channel, all ND messages sent by a symmetric node A are received on time on node B to maintain the symmetric status. As Figure 4.3a shows, node A configures its ND message interval to broadcast a ND message every 2 sec. In node B, ND message hold time expires after 2 sec. In a real transmission channel, interference and collisions delay or kill some packets; when a ND message is not received, node A changes the status of node B to asymmetric neighbor until a new ND message is received as Figure 4.3b shows. During this interval of time, the link is disabled for data traffic and node A is changed to asymmetric to prevent packet loss if this node has moved out of transmission range. When node B receives a new ND message from node A, node B updates its neighbor set table and changes to a symmetric neighbor to node A. The probability to lose a ND message during transmission depends on different *uncontrolled* and *controlled* conditions of congestion, fading signals, noise, etc. As Figure 4.3c shows, when ND message hold time is equal to the ND message interval, node B constantly changes the status of node A from symmetric to asymmetric neighbor in its neighbor set table, and the data transmission becomes unstable between the two nodes, we call this phenomenon *neighbor-flapping*.

Observation 2: The minimum over all packet loss is observed when the ND message hold time approaches to two times the ND message interval; we can see this behavior for all experiments run with different ND values of $\delta = 2, 4, 8, 16$ sec; where the neighbor flapping effect disappears. This is more visible in the individual experiment 1, where there is a topology transition of one link, compared with individual experiments 2, 3 and 4, where more than one link changes in each topology transition.

Observation 3: When the ND message hold time larger than 2 or 3 times the ND message



(a) ND process when δ is equal to τ.

(b) Node B changes to asymmetric neighbor

(c) Node B neighbor flapping

Figure 4.3: Node B neighbor-flapping when δ is equal to τ.

interval; it will take longer to a node to discover when a neighbor is not within its transmission range; and the nodes disseminate invalid links and topology information in the network; this increases the trend of overall packet loss rate for larger ND message hold times.

Observation 4: We can observe different values in the physical testbed and in the NS3-based simulator. In all the experiments the result in the physical test bed is larger than the NS3-based simulator where modeling noise in the transmission is limited; however, we can observe the same trend in the two platforms.

4.3 Analysis

4.3.1 Overall Packet Loss Rate

From the trends on plots 4.1 and 4.2, we can see that when ND message hold time is equal or close to ND message interval, the highest overall packet loss rate occurs. As the ND message hold time approaches to 2 times the ND message interval, the overall packet loss rate values diminishes to its lowest rate. When the ND message hold time increases, the trend of overall packet loss rate increases as well, this is due to nodes taking longer to recognize when a neighbor is no longer within transmission range. Another problem happens for new arriving neighbors, and nodes lose the accuracy of the topology and routing information.

There is a proportional increase of overall packet loss rate when the ND message interval increases from 2 sec, to 4 sec, 8 sec, and 16 sec (seen in plots 4.1), this is due to accuracy of the ND message interval of the mobility simulated in our experiments. This work can be extended to study ND message interval behavior with larger TCI (e.g., 240 sec or 480 sec) that simulate a MANET with lower mobility. ND message intervals of 4 sec or 8 sec are more suitable for conditions of low mobility because it will reduce overhead and congestion in the network. Finally, for the experiments with ND message interval of 32 sec, we can see high overall packet loss rate with all values of ND message hold time. This reflects that the ND message interval when the ND messages are so sparse with regular mobility, the nodes lose the accuracy of the neighbors information, increasing the packet loss.

4.3.2 Neighbor-flapping

As we have explained in the previous section, *neighbor-flapping* phenomenon appears when the ND message hold time is equal to the ND message interval, to prove this, we run a set of 30 experiments to measure the number of changes in the neighbors and the percentage of time neighbors remain asymmetric. In Table 4.1 we show the different tuples of ND parameters used in each experiment. Each experiment lasted for 1440 sec with no topology changes. The topology

followed is as indicated in Figure 4.4, where each node maintains links with different numbers of one-hop neighbors. To measure the impact of data traffic in *neighbor-flapping*, two sets of experiments were run: the first set of experiments without data traffic and the second set with data traffic.

Table 4.1: Tuples of (δ, τ) to measure neighbor-flapping.

τ (sec)	$\delta = 2$ sec	$\delta = 4$ sec	$\delta = 8$ sec
(1.00δ)	2.00	4.00	8.00
(1.10δ)	2.20	4.40	8.80
(1.25δ)	2.50	5.00	10.00
(1.50δ)	3.00	6.00	12.00
(2.00δ)	4.00	8.00	16.00
(2.25δ)	4.50	9.00	18.00
(2.50δ)	5.00	10.00	20.00
(3.00δ)	6.00	12.00	24.00
(4.00δ)	8.00	16.00	32.00
(6.00δ)	12.00	24.00	48.00

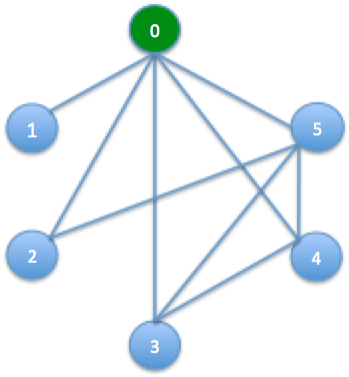
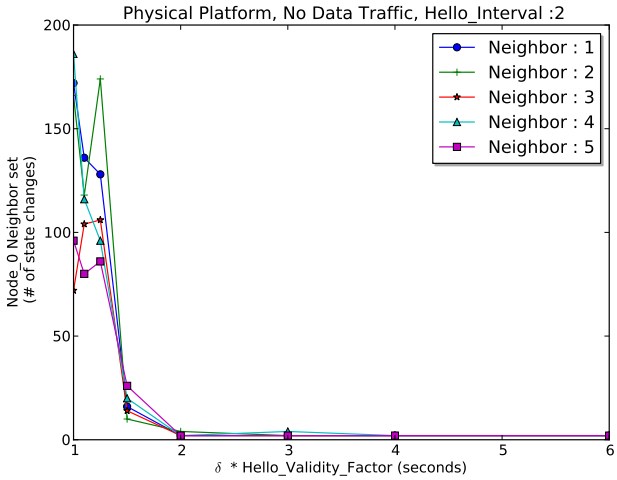
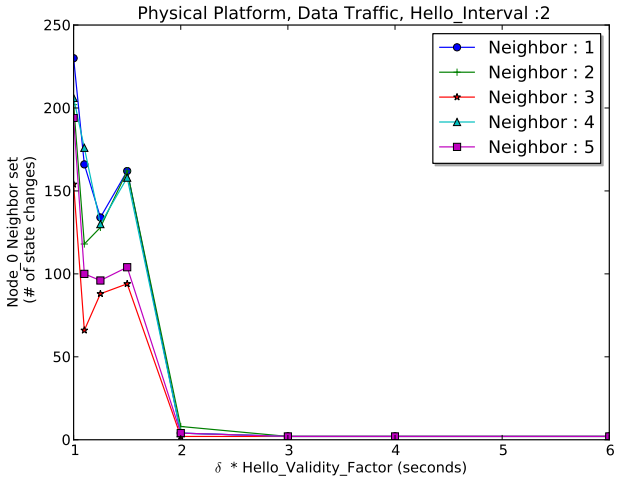


Figure 4.4: Neighbor-flapping topology.

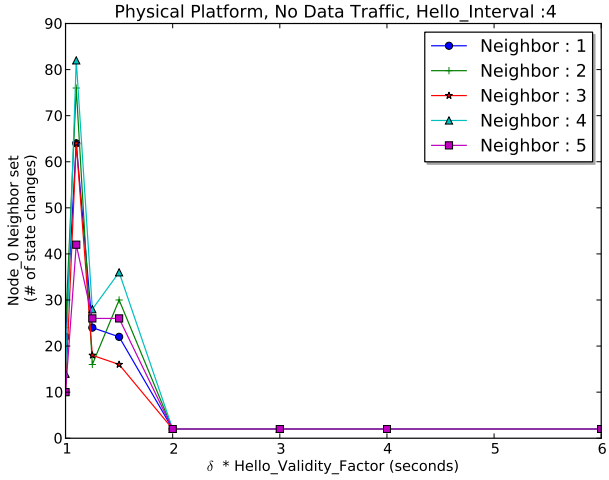
In Plots 4.5 and 4.6 we show the results to measure neighbor-flapping, in the left side is the plot of neighbor-flapping *without data traffic* and in the right is the plot with *data traffic*. As we can see in plots 4.5b), d) and f), the presence of data traffic increases congestion in the transmission channel; more ND messages are lost, and a higher rate of neighbor-flapping results.



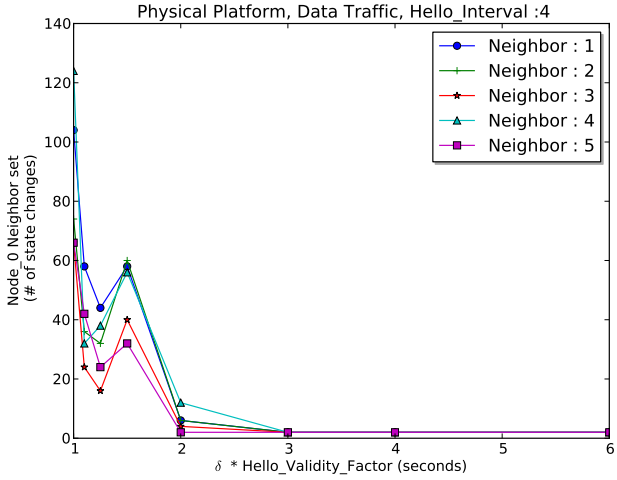
a) (δ) = 2 sec no data traffic



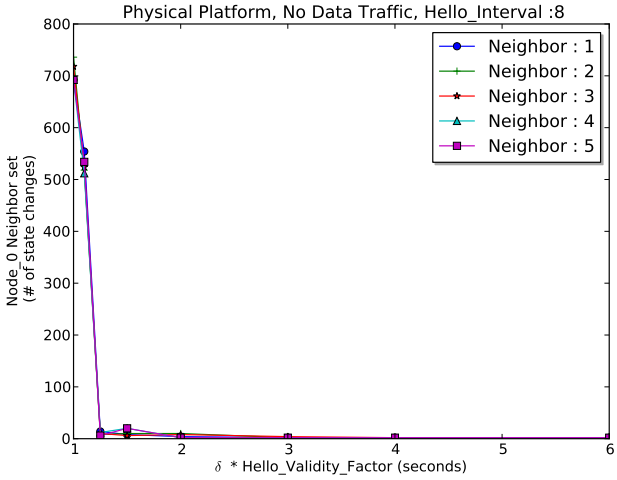
b) (δ) = 2 sec data traffic



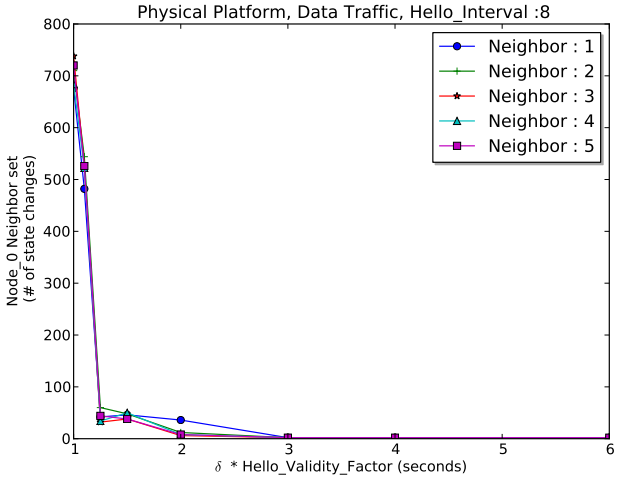
c) (δ) = 4 sec no data traffic



d) (δ) = 4 sec data traffic

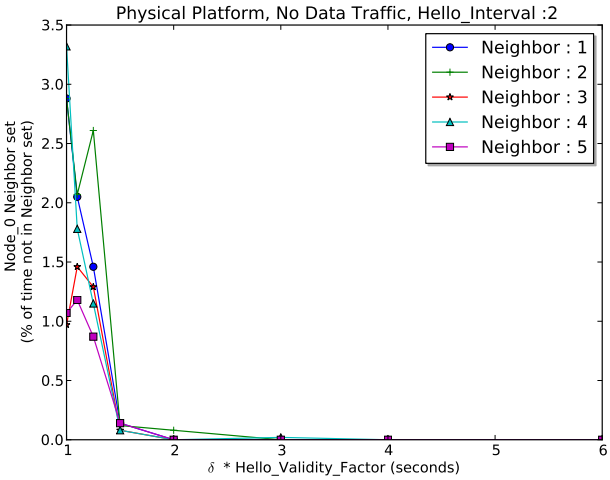


e) (δ) = 8 sec no data traffic

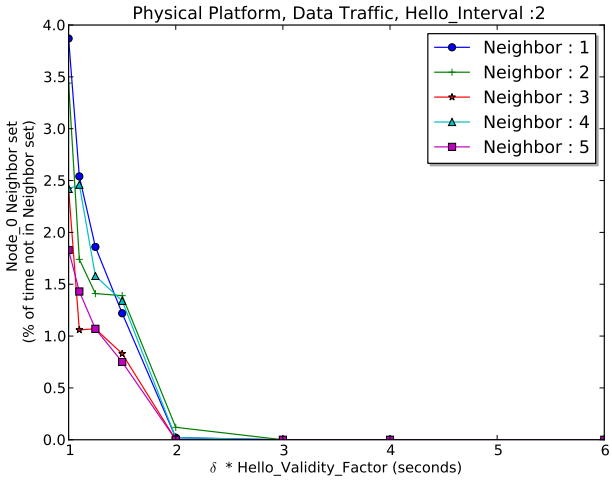


f) (δ) = 8 sec data traffic

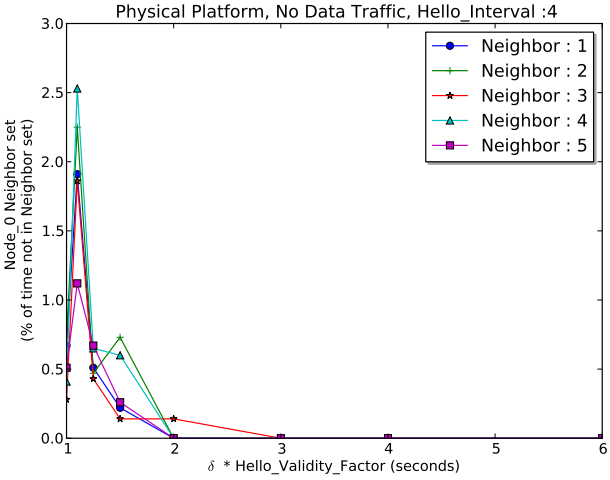
Figure 4.5: Neighbor-flapping comparison



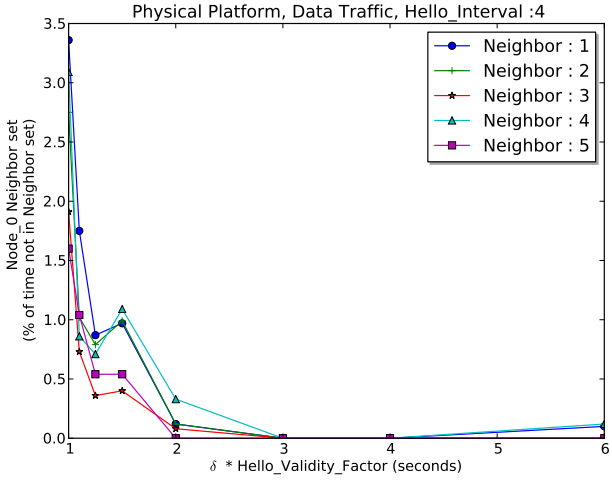
a) (δ) = 2 sec no data traffic



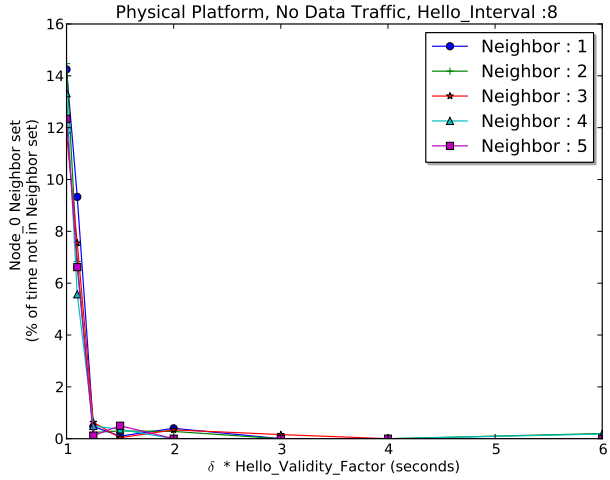
b) (δ) = 2 sec data traffic



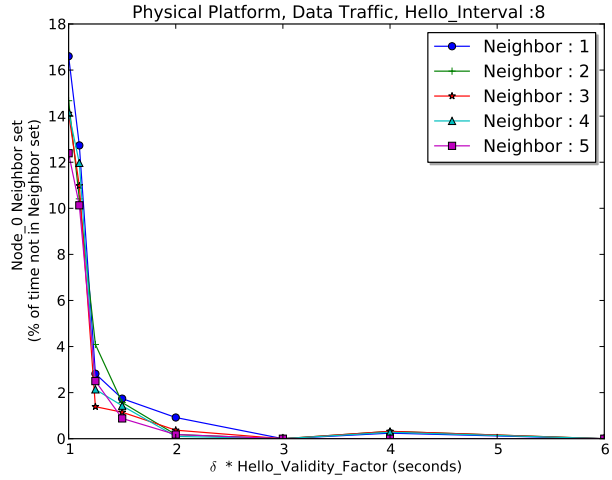
c) (δ) = 4 sec no data traffic



d) (δ) = 4 sec data traffic



e) (δ) = 8 sec no data traffic



f) (δ) = 8 sec data traffic

Figure 4.6: Asymmetric time neighbor-flapping comparison

Chapter 5: Conclusions

5.1 Summary

MANETs are highly flexible and mobile wireless networks suitable for different applications. In order to gain a reliable data transmission, different process such as: Neighbor Discovery (ND), Topology Discovery and Routing Table Calculation must be accomplish to gain a reliable data transmission. In MANETs, Topology Discovery and Routing Table Calculation entirely information is obtained from the ND process. ND process is the periodic exchange of ND messages to discover one-hop neighbors; this process is defined by two parameters: ND message interval (the period of time in between sending the ND messages) and the ND hold times (the validity of the ND messages).

In this work we ran a set of experiments in a pseudo-mobile MANET with six-nodes in a physical testbed; the experiments were run using OLSR routing protocol and a constant bit rate of data traffic. We evaluate different values of ND parameters to measure the impact in the overall packet loss rate. This testing environment was also simulated using a NS3-based simulator to validate the results from the testbed. In our experiments, we found the highest overall packet loss rate when the ND messages hold time is equal to the ND message. The source of this behavior is the neighbor-flapping phenomenon that results of the interference of the increase of traffic in the transmission channel where the symmetric status of one-hop neighbors is falsely lost due to the lack of ND messages to refresh their status.

On the other side of the spectrum, when the ND message hold time is extended to higher than 3 times the ND messages interval, nodes lose accuracy of the neighbor set information maintaining incorrect information of the nodes that are not one-hop neighbors, increasing the packet loss rate. Higher packet loss rate appears when the ND message hold time value increases.

According to the results, when the ND message hold time is 25% to 50% larger than the ND message interval, the overall packet loss rate shows a significant decrease; we also found that when the ND message hold time is twice the ND message interval; the neighbor-flapping

phenomenon disappears and the overall packet loss rate plummeted to its minimum value. The minimum overall packet loss remains steady in the interval where the ND message hold time is two to three times the ND message interval. These results were observed in both the physical testbed and the NS3-based simulator.

5.2 Future work

The results obtained in this work apply for MANETs with a regular mobility and the ND parameters values must be adjusted depending upon the mobility on the MANET. For future work, different conditions can be evaluated:

- Extend experiments different conditions of mobility and interference (e.g., different TCI).
- Expose the experimental design to other performance metrics: scalability, and throughput.
- Extend experiments using different data traffic (e.g., audio streams and multimedia streams)
- Continue the same experiments with other protocols (e.g., DSDV, AODV, TORA, etc).

Bibliography

[1] T. Clausen and P. Jacquet, "Rfc 3626," *Optimized link state routing protocol (OLSR)*, 2003.

[2] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, *Mobile Ad Hoc Networking.* Wiley-Interscience, 2004.

[3] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach.* Addison-Wesley, 2013.

[4] S. Saktar, T. Basavaraju, and C. Puttamadappa, *Ad Hoc Mobile Wireless Networks: Principles, Protocols and Applications.* 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL: CRC Press, 2013.

[5] I. Chlamtac, M. Conti, and J. J.-N. Liu, "Mobile ad hoc networking: imperatives and challenges," *Ad Hoc Networks*, vol. 1, pp. 13 – 64, 2003.

[6] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 1, pp. 1 – 22, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S157087050300043X

[7] X. Hong, K. Xu, and M. Gerla, "Scalable routing protocols for mobile ad hoc networks," *Network, IEEE*, vol. 16, no. 4, pp. 11 – 21, Jul 2002.

[8] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, ser. SIGCOMM '94. New York, NY, USA: ACM, 1994, pp. 234–244. [Online]. Available: http://doi.acm.org/10.1145/190314.190336

[9] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol (OLSR)," 2003, network Working Group. [Online]. Available: https://hal.inria.fr/inria-00471712

[10] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye state routing: a routing scheme for ad hoc wireless networks," in *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, vol. 1, 2000, pp. 70–74 vol.1.

[11] T.-W. Chen and M. Gerla, "Global state routing: a new routing scheme for ad-hoc wireless networks," in *Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on*, vol. 1, Jun 1998, pp. 171–175 vol.1.

[12] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, Feb 1999, pp. 90–100.

[13] D. B. JOHNSON, "The dynamic source routing protocol for mobile ad hoc networks," *draft-ietf-manet-dsr-09.txt*, 2003. [Online]. Available: http://ci.nii.ac.jp/naid/10015168215/en/

[14] R. Dube, C. Rais, K.-Y. Wang, and S. Tripathi, "Signal stability-based adaptive routing (ssa) for ad hoc mobile networks," *Personal Communications, IEEE*, vol. 4, no. 1, pp. 36–45, Feb 1997.

[15] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang, "A wireless hierarchical routing protocol with group mobility," in *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, 1999, pp. 1538–1542 vol.3.

[16] C.-C. Chiang and M. Gerla, "Routing and multicast in multihop, mobile wireless networks," in *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, vol. 2. IEEE, 1997, pp. 546–551.

[17] Z. J. Haas and M. R. Pearlman, "The performance of query control schemes for the zone routing protocol," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 4, pp. 427–438, 2001.

[18] G. Pei, M. Gerla, and X. Hong, "Lanmar: landmark routing for large scale wireless ad hoc networks with group mobility," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing.* IEEE Press, 2000, pp. 11–18.

[19] C. Gomez, D. Garcia, and J. Paradells, "Improving performance of a real ad-hoc network by tuning olsr parameters," in *Computers and Communications, 2005. ISCC 2005. Proceedings. 10th IEEE Symposium on.* IEEE, 2005, pp. 16–21.

[20] S. Demers and L. Kant, "Manets: Perfrormance analysis and management," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, Oct 2006, pp. 1–7.

[21] L. Barolli, M. Ikeda, G. De Marco, A. Durresi, and F. Xhafa, "Performance analysis of olsr and batman protocols considering link quality parameter," in *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on.* IEEE, 2009, pp. 307–314.

[22] A. Medina and S. Bohacek, "Performance modeling of neighbor discovery in proactive routing protocols," *Journal of Advanced Research*, vol. 2, no. 3, pp. 227 – 239, 2011.

[23] Y. Huang, S. N. Bhatti, and D. Parker, "Tuning olsr," in *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on.* IEEE, 2006, pp. 1–5.

[24] A. Tnnesen, T. Lopatic, H. Gredler, B. Petrovitsch, A. Kaplan, S.-O. Tucke, *et al.* (2004) olsrd and adhoc wireless mesh routing daemon. [Online]. Available: http://www.olsr.org

[25] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, 2001, pp. 62–68.

[26] P. . C. Services. The ping page. [Online]. Available: http://www.ping127001.com/pingpage.htm

[27] A. Collette, *Python and HDF5*. O'reilly Media, Inc., 2013.

[28] F. Bai, N. Sadagopan, and A. Helmy, "Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, March 2003, pp. 825–835 vol.2.

[29] nsnam.org. Ns-3. [Online]. Available: https://www.nsnam.org

Curriculum Vitae

Esau Ruiz Gaistardo was born in Mexico City, Mexico. He has experience over 10 years in the Electronic Manufacturing Industry in Mexico; in 2012, he joined the University of Texas at El Paso (UTEP) to pursue a M.S. in Computer Engineering in Electrical Engineering. Upon admission, he has been working in the Network and Communications Lab under the supervision of Dr. Michael McGarry. As a graduate researcher, he helped develop simulation experiments for several routing protocols in Mobile Ad-Hoc Networks. The main focus on his research has been in the simulation and developing of testbeds to measure neighbor discovery process and optimization.

Permanent Address: Esau Ruiz-Gaistardo
6501 McNutt Rd.
Antony, NM. 88021
USA